МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОНЛАЙН РЕАЛІЗАЦІЇ ТОВАРІВ З ОСОБИСТИМ КАБІНЕТОМ КОРИСТУВАЧА»**

Виконав: студент 4 курсу, групи 6.1261
спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)
освітньої програми Інформаційні системи та технології
(назва освітньої програми)
В.Б Дорофеєв
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н. Борю С.Ю
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри програмної інженерії, доцент, к.ф.-м.н. Лісняк А.О
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 126 Інформаційні системи та технології
(шифр і назва)

Освітня програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Шило Г.М.
(підпис)

" ____ " _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Дорофеєву Володимиру Борисовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка інформаційної системи для онлайн реалізації товарів з особистим кабінетом користувача

керівник роботи Борю Сергій Юрійович, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 09.06.2025

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка інформаційної системи для онлайн реалізації товарів з особистим кабінетом користувача.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 20.12.2024

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	02.04.2025	
2.	Збір вихідних даних.	15.04.2025	
3.	Обробка методичних та теоретичних	27.04.2025	
	джерел.		
4.	Розробка першого та другого розділу.	08.05.2025	
5.	Розробка третього розділу.	17.05.2025	
6.	Оформлення та нормоконтроль	02.06.2025	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	20.06.2025	

Студент _____ Дорофеєв В.Б.
(підпис) (ініціали та прізвище)

Керівник роботи _____ Борю С.Ю.
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ Спиця О.Г.
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інформаційної системи для онлайн реалізації товарів з особистим кабінетом користувача»: 81 с., 28 рис., 4 табл., 35 джерел.

АДМІНІСТРУВАННЯ, БАЗА ДАНИХ, ВЕБЗАСТОСУНОК, ВЕБІНТЕРФЕЙС, ІНФОРМАЦІЙНА СИСТЕМА, КОРИСТУВАЧ, МОДЕЛІ DJANGO, ОНЛАЙН-ТОРГІВЛЯ, ОСОБИСТИЙ КАБІНЕТ, ПРОГРАМНА РЕАЛІЗАЦІЯ, СЕСІЯ, СТРУКТУРА САЙТУ, ШАБЛОНИ.

Об’єкт дослідження – процес створення та функціонування інформаційної системи онлайн-торгівлі з підтримкою персоналізованого користувацького простору.

Мета роботи: розробити інформаційну систему для онлайн-продажу товарів з особистим кабінетом користувача.

Метод дослідження – структурний аналіз, моделювання інформаційних потоків, метод об'єктно-орієнтованого програмування, прототипування, застосування шаблонів архітектури MVC, використання фреймворку Django та реляційної моделі баз даних SQLite.

У межах виконання кваліфікаційної роботи було досліджено основи створення інформаційних систем для онлайн-торгівлі. Вивчено актуальні підходи до розробки вебзастосунків, архітектуру MVC та принципи роботи фреймворку Django. Отримані теоретичні знання були реалізовані на практиці у вигляді інформаційної системи з особистим кабінетом користувача, можливістю додавання товарів у кошик та оформлення замовлень. У розробці використано мову програмування Python, HTML, CSS (Bootstrap), а також реляційну базу даних SQLite. Результатом роботи є функціональний вебсайт, що моделює процес онлайн-продажу товарів.

SUMMARY

Bachelor's Qualifying Theses « Development of an Information System for Online Sales of Goods with a Personal User Account»: 81 pages, 28 figures, 4 tables, 35 references.

ADMINISTRATION, DATABASE, WEB APPLICATION, WEB INTERFACE, INFORMATION SYSTEM, USER, DJANGO MODELS, ONLINE TRADE, PERSONAL ACCOUNT, SOFTWARE IMPLEMENTATION, SESSION, SITE STRUCTURE, TEMPLATES.

Object of the study: the process of development and functioning of an information system for online commerce with support for a personalized user space.

Aim of the study: to develop an information system for online product sales that enables users to create personal accounts, browse products, place orders, and manage them through a convenient interface.

Methods of research: structural analysis, modeling of information flows, object-oriented programming, prototyping, MVC architecture patterns, Django framework, and the relational database model SQLite.

As part of the bachelor's qualification work, the fundamentals of developing information systems for online commerce were explored. Modern approaches to web application development, the MVC architecture, and principles of the Django framework were studied. The acquired theoretical knowledge was applied in practice through the creation of an information system with a personal user account, shopping cart functionality, and order placement. The development was implemented using Python, HTML, CSS (Bootstrap), and the relational database SQLite. The result is a functional website that simulates the online sales process.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 8
1 Особливості інформаційних систем в онлайн торгівлі 9
1.1 Роль інформаційної системи в онлайн торгівлі. Особливості використання інформаційних систем в онлайн торгівлі 9
1.2 Переваги ведення онлайн торгівлі за допомогою інформаційних систем 10
1.3 Основні цілі та задачі, які вирішуються особистим кабінетом користувача 11
1.4 Огляд існуючих інформаційних систем в торгівлі 12
1.4.1 Vans – офіційний інтернет-магазин американського виробника взуття та одягу 12
1.4.2 PlayStation – офіційний інтернет-магазин ігрового гіганту 15
1.4.3 Порівняння оглянутих інформаційних систем 19
1.5 Мета роботи 20
2 Розробка інформаційної системи для онлайн реалізації товарів з особистим кабінетом користувача 22
2.1 Виявлення особливостей інформаційної системи для онлайн реалізації товару 22
2.1.1 Косметика 22
2.1.2 Одяг 24
2.1.3 Біжутерія 25
2.2 Розробка вимог до інформаційної системи 26
2.2.1 Функціональні вимоги для інформаційної системи 27

2.2.2 Нефункціональні вимоги для інформаційної системи 29
2.3 Формалізація основних задач інформаційної системи 30
2.4 Проєктування архітектури інформаційної системи 32
2.4.1 Монолітна архітектура 33
2.4.2 Клієнт-серверна архітектура 34
2.4.3 Архітектура MVC 35
2.5 Розробка інформаційної моделі інформаційної системи 36
2.6 Розробка структури інформаційної системи 39
2.2 Висновки до розділу 43
3 Програмна реалізація інформаційної системи для онлайн реалізації товарів з особистим кабінетом користувача 44
3.1 Вибір засобів реалізації інформаційної системи 44
3.2 Структура програмного проєкту 47
3.2.1 Конфігурація проєкту 52
3.2.2 Шаблони та статичні файли 53
3.2.3 Робота з базою даних 54
3.3 Реалізація функціоналу користувача 55
3.3.1 Реєстрація та авторизація користувача 55
3.3.2 Особистий кабінет користувача 58
3.3.3 Захист доступу до персональних сторінок 61
3.4 Реалізація каталогу товарів 63
3.4.1 Структура моделі товару 63
3.4.2 Категорії товарів та фільтрація 65
3.4.3 Сторінка товару 67
3.5 Реалізація кошика та оформлення замовлення 70
3.6 Висновки до розділу 3 75
Висновки 77
Перелік посилань 78

ВСТУП

Сучасні тенденції цифрової трансформації суттєво впливають на сферу торгівлі, сприяючи активному розвитку електронної комерції. Інтернет-магазини поступово витісняють традиційні канали продажів, надаючи користувачам зручний доступ до товарів і послуг незалежно від географічного розташування чи часу доби. Це створює потребу у розробці функціональних, стабільних і зручних інформаційних систем, здатних забезпечити повноцінну підтримку онлайн-продажів.

Одним із важливих аспектів створення таких систем є забезпечення інтуїтивно зрозумілого інтерфейсу, надійної авторизації користувачів, підтримки обліку замовлень, а також можливості керування даними через особистий кабінет. Усе це потребує не лише продуманої архітектури, а й використання сучасних технологій розробки веб-застосунків.

Актуальність теми обумовлена зростаючим попитом на автоматизацію процесів онлайн-продажу та підвищення рівня обслуговування клієнтів шляхом впровадження спеціалізованих веб-рішень.

Мета даної кваліфікаційної роботи — розробити інформаційну систему для підтримки онлайн-продажу товарів із функціоналом персонального кабінету користувача. У процесі роботи буде проаналізовано вимоги до такої системи, спроєктовано її архітектуру та реалізовано ключові компоненти.

Об'єктом дослідження є процес функціонування інформаційних систем у сфері електронної комерції. Предметом — програмна реалізація веб-застосунку для онлайн-замовлень.

1 ОСОБЛИВОСТІ ІНФОРМАЦІЙНИХ СИСТЕМ В ОНЛАЙН ТОРГІВЛІ

1.1 Роль інформаційної системи в онлайн торгівлі. Особливості використання інформаційних систем в онлайн торгівлі

Інформаційна система (ІС) – це сукупність організаційних і технічних засобів для збереження та обробки інформації з метою забезпечення інформаційних потреб користувачів [1].

У контексті електронної комерції інформаційна система забезпечує стабільне функціонування онлайн-магазину від етапу перегляду їх змісту клієнтом до фінального етапу оформлення і доставки замовлення.

В онлайн торгівлі інформаційна система реалізує такі функції:

- автоматизація ведення звітності та аналітики – полегшує збір статистики, що стосується продажів, поведінки та інтересів клієнтів, аналіз цільової аудиторії; на основі цієї інформації можна редагувати асортимент, ціноутворення, маркетингову політику;
- управління товарами – швидке додавання, редагування, видалення товару, зміна ціни або встановлення акцій;
- обробка замовлень – автоматичне прийняття замовлення, інформування клієнта про статус покупки.

Автоматизація процесів за допомогою інформаційних систем має важливе значення для ефективності продажів. ІС дозволяє значно зменшити об’єм ручної роботи, мінімізує кількість помилок, пришвидшує обробку замовлень та покращує рівень взаємодії з клієнтом. Завдяки цьому підприємець може сконцентрувати увагу не на рутинних процесах, а на розвиток бізнесу, маркетингу та покращенні клієнтського досвіду.

1.2 Переваги ведення онлайн торгівлі за допомогою інформаційних систем

Використання ІС у сфері онлайн-торгівлі дає змогу підприємствам підвищити рівень ефективності ведення бізнесу, допомагає автоматизувати ключові бізнес-процеси та створити зручний і привабливий інтерфейс користувача для клієнтів. Завдяки використанню ІС, навіть невеликий онлайн-магазин може працювати на одному рівні із великими маркетплейсами, використовуючи інструменти, доступ до яких був відкритий тільки для великих компаній.

До основних переваг онлайн-торгівлі з використанням ІС можна віднести:

- цілодобовий доступ для клієнтів – на відмінну від класичних офлайн магазинів, в інтернет-магазинах система цілодобово без участі продавця, що розширює можливості для продажів;
- зниження операційних витрат – немає потреби в оренді високовартісного приміщення, великому складі або численному персоналі;
- можливість персоналізації – за допомогою сучасних алгоритмів система може відстежувати поведінку та інтереси клієнта для того, щоб показувати відповідні рекомендації та пропозиції;
- автоматизація бізнес-процесів – інформаційна система надає можливість частково або повністю оптимізувати облік замовлень, контроль залишків, генерацію звітності та підтримку клієнтів.

Отже, інформаційні системи створюють технологічну основу для стабільного розвитку електронної комерції, дозволяючи бізнесу бути більш гнучким, ефективним і конкурентоспроможним в поточному ринку.

1.3 Основні цілі та задачі, які вирішуються особистим кабінетом користувача

Особистий кабінет користувача – це елемент інтерфейсу сайту, який робить для користувача досвід більш персоналізованим. Завдяки наявності даної опції, користувачі отримують доступ до функціоналу, характерному саме для їх власних акаунтів [2].

Особистий кабінет користувача є важливою складовою будь-якого сучасного інтернет-магазину. Його головна мета – зробити взаємодію користувача із системою максимально зручною, швидкою та персоналізованою. Завдяки кабінету користувач отримує можливість відстежувати свої замовлення, переглядати статус доставки, отримувати важливі сповіщення та новини. Це створює в клієнта відчуття довіри до магазину та впевненості під час користування сайтом.

Завдяки наявності особистого кабінету покупець може неодноразово повертатись на сайт без потреби кожного разу вводити свої особисті дані або шукати товар чи замовлення вручну. Вся інформація зберігається в одному місці: історія покупок, улюблені товари, збережені адреси для доставки та інші дані, необхідні для оформлення замовлення. Така функціональність не лише полегшує процес покупки, а й формує позитивний клієнтський досвід, адже користувач бачить, що система «пам’ятає» його попередні дії, надає актуальну інформацію та дозволяє при потребі швидко отримати допомогу.

Окрім зручності для самого користувача, особистий кабінет є дуже цінним інструментом для самого магазину. Інформацію, яку залишає клієнт у своєму профілі та під час покупок, адміністрація магазину отримує змогу проводити аналітику та інтереси цільової аудиторії, частоту покупок, вподобання щодо категорій товарів, рівень залученості користувачів до акцій або розсилок. Такі данні дають можливість ефективніше розуміти поведінку клієнтів і приймати влучні рішення щодо оновлення асортименту, впровадження нових пропозицій або поліпшення навігації по сайту. Наприклад, якщо система фіксує, що

більшість користувачів додають улюблені товари до списку бажань, але не завершують покупку, це може свідчити про потребу в акції або спеціальній пропозиції. Аналогічно, завдяки накопиченим даним з особистих кабінетів можна формувати персоналізовані пропозиції, створювати ефективні розсилки або класифікувати клієнтів за рівнем активності або смаковими вподобаннями.

Таким чином, особистий кабінет стає не тільки зручним елементом для клієнта, а й важливою аналітичною складовою, яка дозволяє бізнесу адаптуватися до потреб клієнтів, покращувати якість сервісу та підвищувати продажі.

1.4 Огляд існуючих інформаційних систем в торгівлі

В сучасному світі багато компаній та підприємств, які так чи інакше пов’язані із сферою продажів мають свої інформаційні системи для продажів в інтернеті. Для аналізу розглянемо 2 сучасні веб-ресурси електронної комерції, що представляють онлайн-магазини відомих брендів: Vans та Playstation. Для аналізу та подальшого порівняння застосуємо такі критерії, як: наповненість головної сторінки, наявність особистого кабінету користувача та його можливості (перегляд замовлень, редагування особистої інформації), пошук та фільтрація товару, наявність функції «Додати в список побажань» а також додаткові спеціальні можливості.

1.4.1 Vans – офіційний інтернет-магазин американського виробника взуття та одягу

Vans – відомий бренд молодіжного взуття і одягу, чий офіційний сайт пропонує широкий асортимент продукції. Одразу після переходу на сайт користувача зустрічає головна сторінка (див. рис. 1.1), на якій можна побачити такі елементи як: контекстне меню в лівому верхньому куту, логотип бренду по

центру зверху, що веде на головну сторінку сайту.

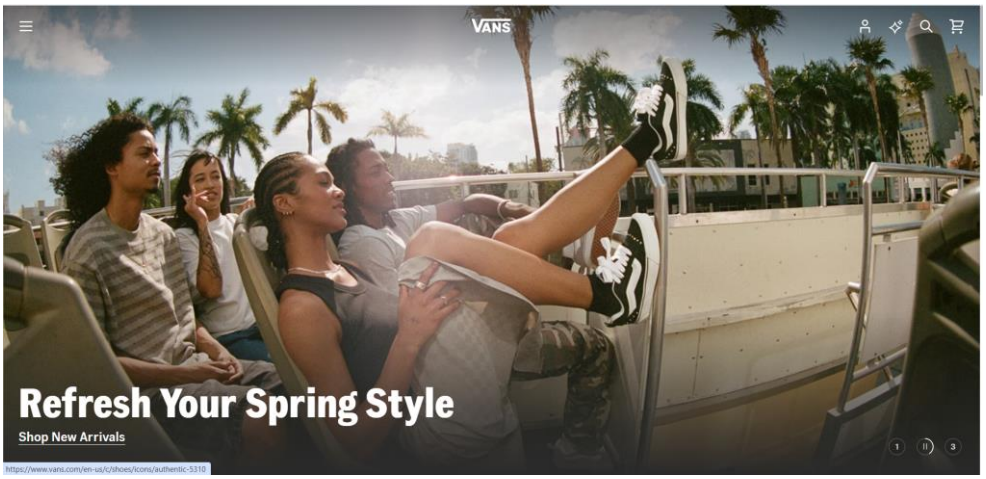


Рисунок 1.1 – Головна сторінка Vans

У верхньому правому куту бачимо такі елементи: особистий кабінет користувача, AI-помічник, пошук по сайту та кошик. Більшу частину головного меню займають інформаційні банери з назвами колекцій, які ведуть на сторінку із переліком товарів.

Щоб увійти до персонального кабінету користувач натискає на логотип профілю у шапці сайту. На рисунку 1.2 зображено інтерфейс кабінету користувача даного сайту.

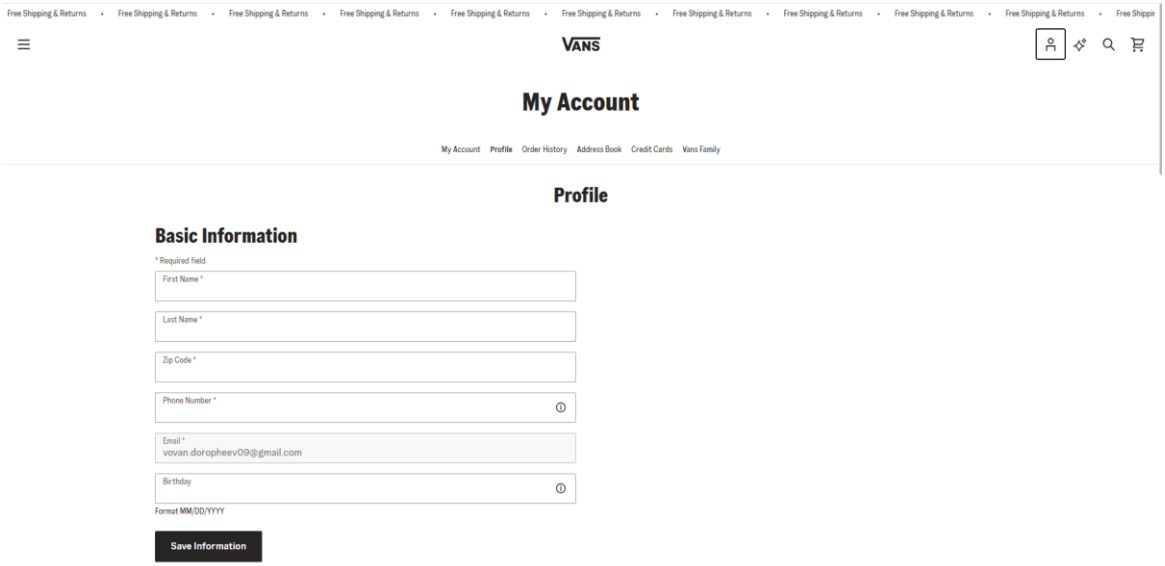


Рисунок 1.2 – Інтерфейс кабінету користувача Vans

Будь який зареєстрований клієнт може переглядати й відстежувати історію замовлень (Order History), додавати та зберігати декілька адрес доставки (Adress Book) та банківських карт (Credit Cards), а також налаштовувати особисті дані профілю. Також через особистий кабінет реалізовано можливість потрапити на сторінку програми лояльності Vans Family – що є великою перевагою магазину перед клієнтом порівняно з конкурентами.

Для аналізу можливості фільтрувати товар перейдемо до списку товарів (див. рис. 1.3).

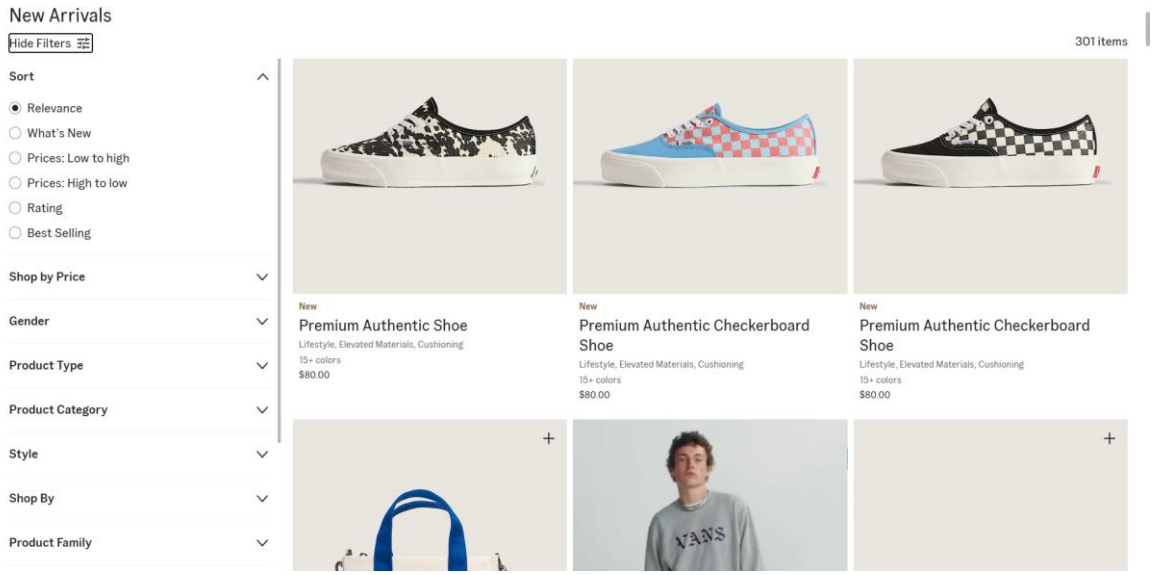


Рисунок 1.3 – Каталог Vans

Система надає великий список можливостей для фільтрації товару: за популярністю, новизною, ціною – від низьких до високих, і навпаки. Також реалізовано можливість шукати товар в певному ціновому діапазоні. Окрім цих параметрів користувач може сортувати товар за такими критеріями як: стиль продукції, колір, розмір, колекція, матеріал, вік дітей та активність – із чого можна зробити висновок, що даний сайт має просунуту можливість фільтрації товарів.

Натиснувши на будь-який продукт, система перенаправляє користувача зі сторінки показу товарів на сторінку товару (див. рис. 1.4).

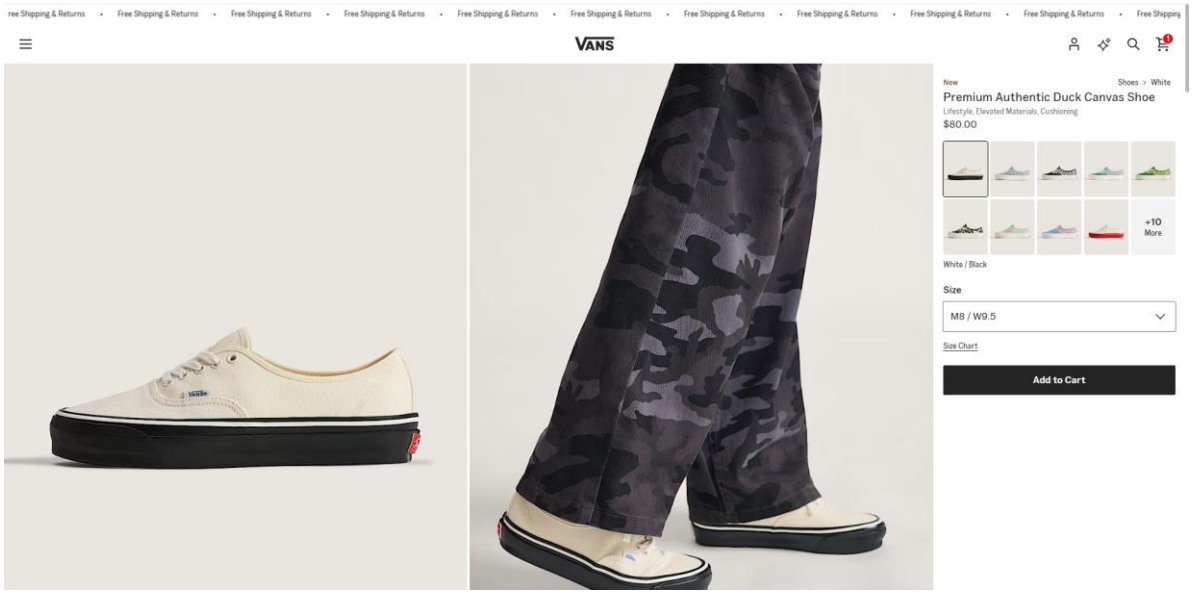


Рисунок 1.4 – Сторінка товару Vans

На сторінці товару користувач має можливість обрати дизайн товару, що йому найбільше подобається, а також обрати необхідний розмір і додати його в кошик. Однак, система не дає можливості користувачу додати товар до списку побажань – що на мою думку є явним недоліком, адже цей маркетинговий підхід змушує користувача не забути про продукт, а продавцю товару надає цілий ряд переваг, такі як: зменшення кількості покинутих кошиків, спрощення процесу купівлі та утримання потенційних клієнтів [4].

1.4.2 PlayStation – офіційний інтернет-магазин ігрового гіганту

PlayStation – це офіційний онлайн-магазин японської компанії Sony, в якому реалізуються ігрові консолі PlayStation, фірмові аксесуари та ігри. Перейти на сайт можна за посиланням – www.playstation.com . На головній сторінці сайту (див. рис. 1.5) можна спостерігати такі елементи інтерфейсу: вхід до акаунту, пошук по сайту, навігаційне меню в верхній частині сайту з основними розділами, яскравий банер з пропозицією ігор, нижче якого розташовано карусель з актуальними іграми, подіями та акційними пропозиціями.

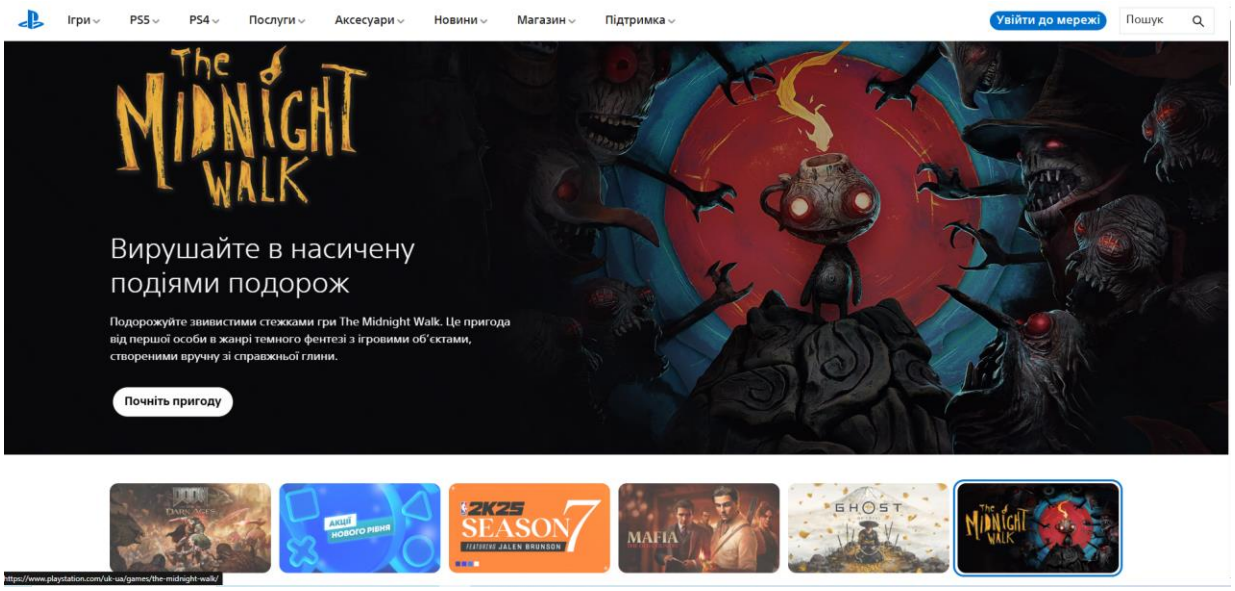


Рисунок 1.5 – Головна сторінка PlayStation

Для входу в особистий кабінет натиснемо «Увійти до мережі», після чого користувача зустрічає сторінка входу (див. рис. 1.6).

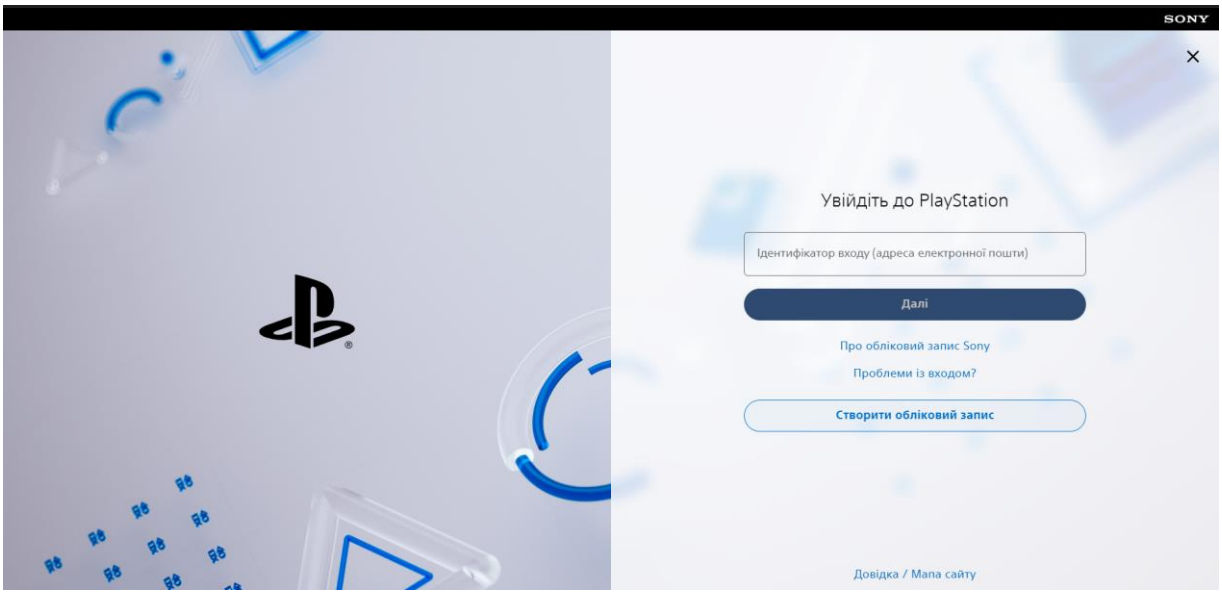


Рисунок 1.6 – Сторінка входу PlayStation

Після вводу даних для входу в акаунт користувача перенаправляє із сайту PlayStation за посиланням https://id.sonyentertainmentnetwork.com/ , яке виступає в ролі особистого кабінету користувача (див. рис. 1.7) . Це означає, що користувач не створює окремий акаунт для PlayStation Store, а входить за

допомогою уніфікованого акаунту Sony, який використовується для всіх сервісів Sony – консолей, підписок, ігор, покупки фільмів, музики тощо.

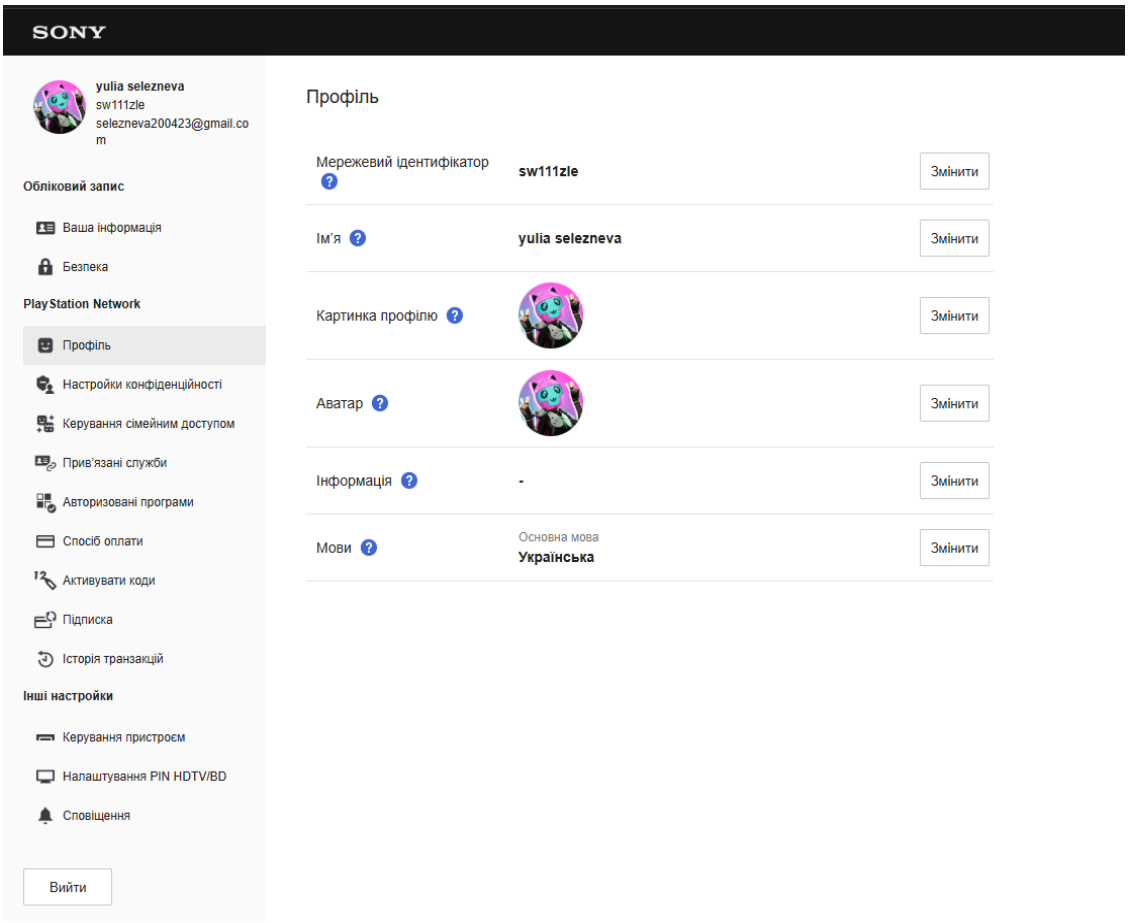


Рисунок 1.7 – Інтерфейс кабінету користувача Sony PlayStation

В особистому кабінеті користувача зібрано всі налаштування облікового запису, які охоплюють як особисту інформацію, так і ігрові та платіжні параметри. На вкладці «Профіль» користувач має змогу передивлятись та редагувати мережевий ідентифікатор, ім’я, картинку профілю та аватар, додавати інформацію про себе та змінювати мову. Особистий кабінет користувача Sony PlayStation також надає можливість керувати підписками, передивлятись історію транзакцій та замовлень, додавати або змінювати способи оплати, або оплачувати кодами (вкладка «Активувати коди»), керувати взаємодією з іншими програмами або сервісами (вкладки «Авторизовані програми» та «Прив’язані служби»). Особистий кабінет також надає можливість користувачу керувати просунутими налаштуваннями, такими як налаштування

безпеки, конфіденційності та налаштування пристроїв.

Для того, щоб оглянути можливості фільтрації товару за різними категоріями, перейдемо до списку товарів. Для того, щоб передивитись товар, необхідно обрати необхідну платформу та вибрати «Ігри», після чого користувача перенаправляє на каталог ігор (див. рис. 1.8).

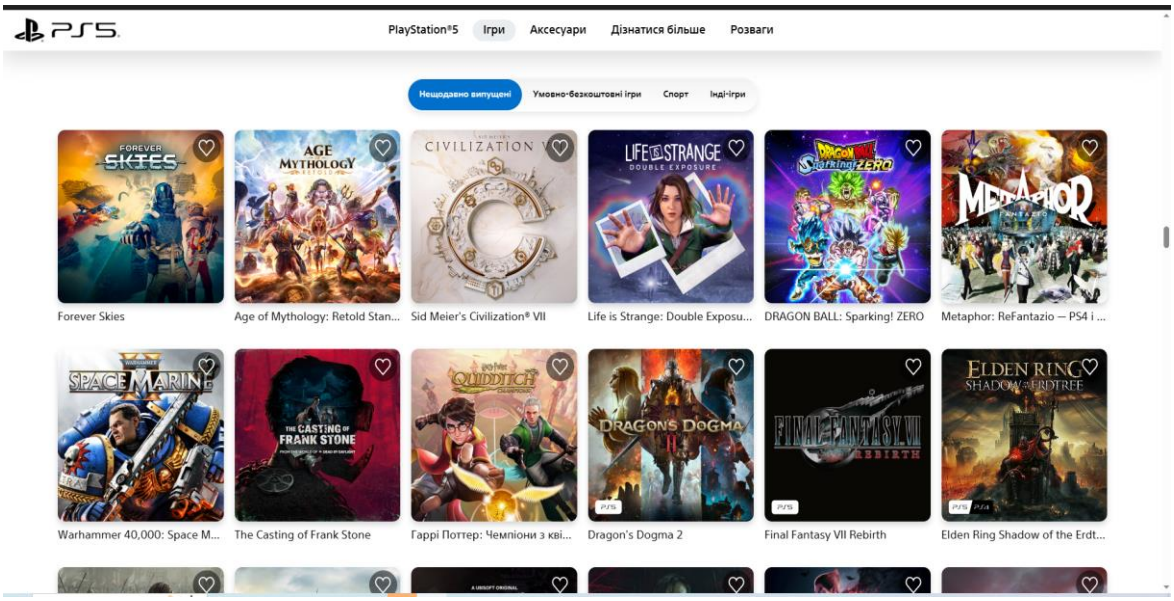


Рисунок 1.8 – Каталог ігор PlayStation

Система надає можливість фільтрувати товар тільки за 4 категоріями: нещодавно випущені, умовно-безкоштовні ігри, спорт та інді-ігри. Це можна вважати певним функціональним обмеженням, оскільки відсутні більш гнучкі параметри фільтрації, такі як рейтинг гри, мова локалізації або наявність багатокористувацького режиму. Для інтернет-магазину, що реалізує велику кількість цифрових товарів, розширені можливості фільтрації товару є критично важливими, адже вони підвищують зручність пошуку, скорочують час взаємодії з каталогом і покращують користувацький досвід.

Перейшовши на будь-яку гру із каталогу, потрапляємо на головну сторінку товару (див. рис. 1.9).

На головній сторінці товару можна спостерігати основні характеристики та можливості гри: кількість гравців, наявність внутрішньо-ігрових покупок,

наявність мережевої гри, вікові обмеження та наявність додаткових функцій. Користувач має можливість обрати пробну версію гри, а також додати гру до «Списку бажань».

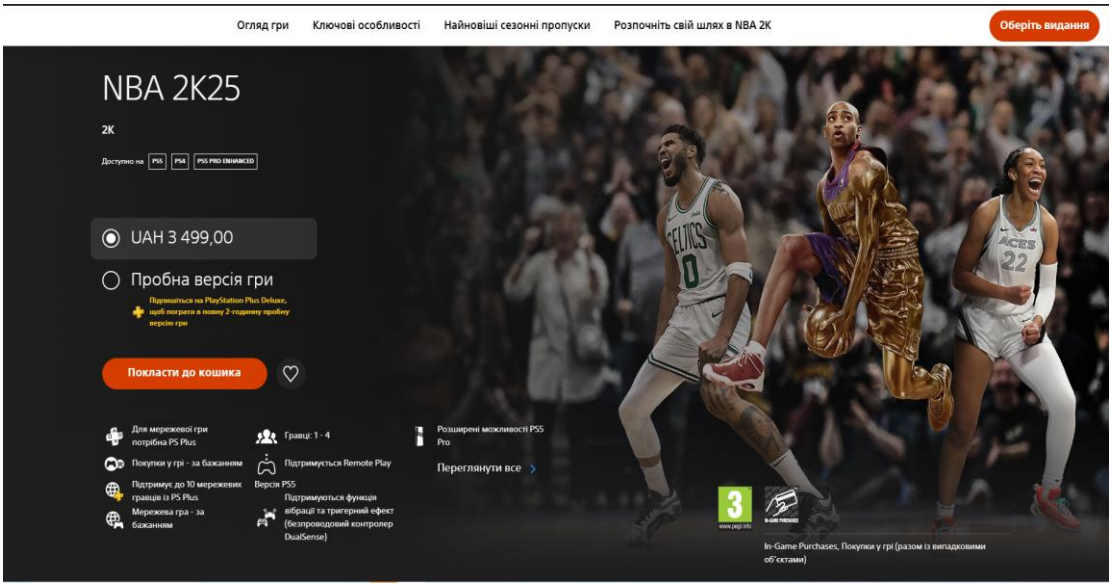


Рисунок 1.9 – Головна сторінка товару PlayStation

1.4.3 Порівняння оглянутих інформаційних систем

На основі проведеного огляду інформаційних систем Vans та PlayStation можна виділити перелік спільних функціональних рішень, а також окремі відмінності у реалізації особистих кабінетів, фільтрації товарів та загальної зручності використання. Порівняння двох ІС наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння оглянутих інформаційних систем

Критерії	Vans	PlayStation
Пошук товару	Реалізовано	Реалізовано
Наповненість головної сторінки	Головна сторінка наповнена банерами із популярними товарами та спеціальними пропозиціями	Головна сторінка наповнена банерами із популярними товарами та спеціальними пропозиціями

Продовження таблиці 1.1

Наявність особистого кабінету користувача	Реалізовано	Реалізовано
Перегляд замовлень	Реалізовано	Реалізовано
Можливість редагування особистої інформації	Реалізовано	Реалізовано
Фільтрація товару	Має розширені можливості фільтрації товарів	Має обмежені можливості фільтрації товарів
Наявність функції «Додати в список бажань»	Не реалізовано	Реалізовано
Додаткові спеціальні можливості	Програма лояльності «Vans Family», AI-асистент	Відсутні

Обидві розглянуті інформаційні системи відповідають базовим вимогам до сучасних e-commerce платформ. Проте, в кожної із систем присутні недоліки, які можна усунути. Наприклад, реалізація функціоналу «Wishlist» допоможе компанії Vans краще утримувати клієнта, зменшити кількість покинутих кошиків та покращити користувацький досвід, а впровадження багатофакторної фільтрації з можливостю комбінацій параметрів значно покращить зручність навігації та полегшить пошук потрібного продукту користувачам магазину PlayStation.

1.5 Мета роботи

Метою роботи є розробка інформаційної системи для онлайн-реалізації товарів з особистим кабінетом користувача.

Для досягнення поставленої мети в роботі вирішуються наступні завдання:

– вибір та аналіз предметної області та виявлення особливостей інформаційної системи для онлайн-реалізації товару обраної предметної області;

- розробка вимог до інформаційної системи;
- формалізація основних задач інформаційної системи;
- проєктування архітектури інформаційної системи;
- розробка інтерфейсу користувача;
- розробка алгоритмів реалізації задач інформаційної системи;
- вибір інструментальних засобів для реалізації програмного продукту;
- розробка системи класів (бібліотек, програм, допоміжних утиліт, SQL-запитів);
- опис основних модулів програмного продукту.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОНЛАЙН РЕАЛІЗАЦІЇ ТОВАРІВ З ОСОБИСТИМ КАБІНЕТОМ КОРИСТУВАЧА

Проєктування інформаційної системи є одним із головних етапів у процесі її розробки. Під час проєктування ІС визначаються її структура, функціональні можливості та логіка взаємодії користувача із програмним продуктом. Це дуже важливий етап, адже від якості його виконання залежить ефективність подальшої реалізації системи та відповідність поставленим вимогам.

У цьому розділі послідовно буде розглянуто вибір та аналіз предметної області, формалізацію функціональних вимог, побудову інформаційної моделі, архітектурне проєктування, а також ключові аспекти, пов’язані із реалізацією інтерфейсу користувача та алгоритмів роботи системи.

2.1 Виявлення особливостей інформаційної системи для онлайн реалізації товару

При розробці інформаційної системи для онлайн-реалізації товару важливо правильно обрати предметну область – тобто категорію товарів, які будуть на ній реалізовуватись. У кожної категорії продукції є свої властивості та вимоги, що впливають на побудову інформаційної системи та стратегію продажів через Інтернет. Серед сфер для розробки успішного інтернет-магазину розглянемо такі напрямки, як косметика, одяг та біжутерія.

2.1.1 Косметика

Сфера косметичних товарів характеризується високим попитом, але й високим рівнем конкуренції з боку великих роздрібних мереж. Ринок косметики контролюють всеукраїнські ритейлери, наприклад Eva та Prostor, які активно

працюють як офлайн, так і онлайн, пропонуючи широкий асортимент продукції, великі знижки та програми лояльності [5]. Новому інтернет-магазину важко змагатися із такими гравцями як за популярністю бренду, так і за ціною. Також косметика належить до категорії товарів, які покупці бажають бачити й тестувати перед купівлею: неможливість відчути аромат чи підібрати відтінок засобу через інтернет спонукає значну частину клієнтів обирати традиційні офлайн-крамниці [6]. Ще одним стримуючим фактором є високовартісний етап логістики до кінцевого покупця: якщо клієнт збирається придбати невелику кількість товару, або економія від онлайн-замовлення не перевищує вартість доставки, то він скоріше придбає потрібний товар у найближчому магазині, де можна отримати товар миттєво, не витрачаючи кошти і час на доставку. Отже, якщо розглядати нішу косметики, то ті переваги, які вона має (великий та сталий попит) перекриваються недоліками, які являють собою суттєві обмеження для розвитку саме онлайн-продажів.

У процесі проєктування інформаційної системи для торгівлі косметикою особливої уваги потребує модель даних, що враховує специфіку продукції. Для забезпечення ефективного пошуку, фільтрації та подання товарів у системі необхідно зберігати параметри, за якими користувач здійснює вибір. Серед ключових характеристик, що повинні бути реалізовані в базі даних:

- назва товару;
- тип продукту;
- виробник або бренд;
- об’єм;
- колір;
- призначення;
- ціна;
- категорія;
- наявність акцій.

2.1.2 Одяг

Індустрія одягу так само насичена відомими брендами і роздрібненими мережами магазинів по всьому світу та Україні. По всій країні у великих торговельних центрах представлені магазини всесвітньо відомих брендів – Zara, Sinsay, Bershka, H&M. Ці мережі забезпечують покупцям легкий сервіс та постійний доступ до актуального одягу як в офлайн, так і в онлайн-режимі. Для прикладу, польський бренд Sinsay тільки протягом 2023 року збільшив кількість своїх магазинів в Україні з 55 до 101. У 2024 році ритейлер в середньому відкривав ще 6-7 магазинів на місяць [7]. Жорстока офлайн-конкуренція також доповнюється особливістю даної категорії товарів: одяг важливо приміряти перед покупкою, щоб впевнитись у відповідності розміру, якості матеріалу, зовнішньому вигляді та комфорті. Онлайн-формат позбавлений цієї переваги, через що замовлення одягу в інтернет-магазині завжди несе ризик розчарування через невідповідність розміру чи інших очікувань, а отже – додаткові клопоти, пов’язані із довгим процесом повернення товару та коштів. Щоб частково компенсувати ці труднощі, інформаційна система для онлайн торгівлі повинна опрацьовувати та зберігати ряд параметрів, які допоможуть користувачу зробити більш обґрунтований вибір. До таких характеристик варто включити:

- назва товару;
- тип одягу;
- розмір;
- стать;
- колір;
- матеріал;
- бренд;
- сезонність;;
- ціна;
- наявність акції.

Із всього вищесказаного можна зробити висновок, що попри значний ринок і попит, дана категорія товару має значні недоліки для продажу товарів в онлайн-форматі: висока конкуренція з боку офлайн-гігантів і вагання клієнтів щодо дистанційної покупки без попередньої примірки і ризику розчарування.

2.1.3 Біжутерія

Сегмент біжутерії (доступних прикрас із недорогоцінних матеріалів) вирізняється особливостями, що роблять його привабливим для онлайн-торгівлі. Попит на біжутерію в Україні стабільно високий, і значна його частка вже реалізується через інтернет: за оцінками експертів, близько 60% продажів товару даної категорії здійснюється саме онлайн, що пояснюється зручністю та широким асортиментом прикрас в мережі [8]. Біжутерія – це товар невеликого розміру і високої цінності за одиницю ваги, тому її зручно зберігати та відправляти поштою – доставка не потребує значних витрат чи складних умов зберігання. Вагомою перевагою невеликого розміру продукції є можливість швидко перевозити склад продукції, що дуже важливо в умовах військового стану, адже часто підприємці становляться заручниками становища через неможливість перевезти свій бізнес із небезпечних умов. Ще одним плюсом біжутерії є яскравий вигляд, який полегшує маркетинг та просування продукції: в порівнянні з одягом, який потребує примірки, прикраси можна ефектно презентувати на фотографіях, адже їх зовнішній вигляд є ключовою характеристикою товару [9]. Мінімальним є ризик того, що клієнт відмовиться від товару через невідповідність розміру, адже більшість прикрас є універсальними за рахунок того, що розмір регулюється.

Інформаційна система онлайн-магазину біжутерії повинна враховувати специфіку товару та передбачити збереження таких основних атрибутів прикрас у базі даних:

- назва товару;
- категорію товару;
- ціна;

- акційна ціна (за наявності акції);
- опис товару;
- характеристика товару.

Ці атрибути повинні використовуватись для фільтрації, пошуку, сортування та відображення товарів на сайті, забезпечуючи зручність користування як клієнтам, так і адміністраторам сайту.

Варто відзначити і структуру ринку біжутерії: окрім кількох великих мереж на кшталт Pandora чи SOVA, які пропонують класичні колекції прикрас, існує безліч невеликих виробників, майстрів та ювелірів, які створюють унікальні товари ручної роботи. Такі авторські вироби часто важко знайти в офлайн-магазинах, натомість через інтернет вони швидко знаходять своїх поціновувачів [10]. Унікальні напрямки, наприклад, етнічна біжутерія або мінімалістичні аксесуари чи авангардні дизайнерські речі, це ті товари, які здатні знайти свого клієнта по всій країні, чого важко досягти за допомогою офлайн-магазину в межах одного, навіть великого міста.

Отже, з огляду на порівняння, найбільш ефективною та перспективною предметною областю для розробки інформаційної системи та подальшого користування нею було обрано область біжутерії. Саме для цієї області найбільше підходить реалізація товару в онлайн-форматі через відсутність великих офлайн-конкурентів, низьких ризиків повернення товару та широку географію клієнтів. В результаті, саме біжутерія видається найбільш оптимальним вибором для успішної онлайн-реалізації товарів.

2.2 Розробка вимог до інформаційної системи

Вимоги до інформаційної системи є одним із ключових понять у розробці програмного забезпечення. Термін «вимоги» в контексті розробки програмного забезпечення означає формальне визначення характеристик та функцій, які система забезпечує для задоволення користувацьких потреб, а також інших зацікавлених сторін [11]. Іншими словами, саме вимоги описують те, що

очікується від інформаційної системи і які властивості вона повинна мати. Без належно сформульованих вимог неможливо ефективно спроєктувати та реалізувати систему, саме тому розробка вимог є важливим етапом розробки.

До інформаційних систем прийнято розрізняти такі вимоги, як функціональні та нефункціональні (див. рис. 2.1) [12].

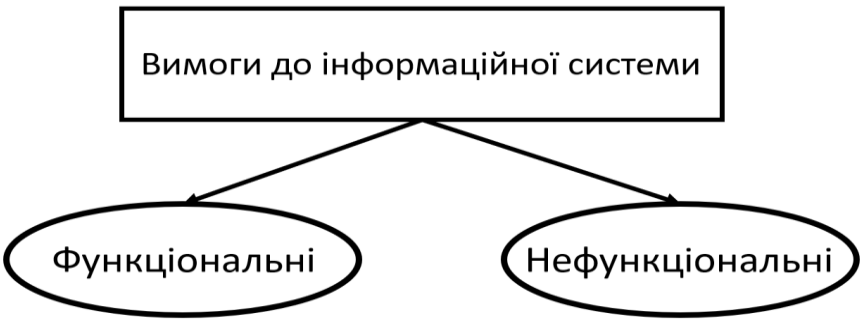


Рисунок 2.1 – Вимоги до інформаційної системи

Для інформаційної системи для онлайн-реалізації товарів з особистим кабінетом користувача визначимо як функціональні, так і нефункціональні вимоги з урахуванням специфіки та особливостей обраної предметної області. Система матиме дві основні ролі користувачів – клієнт (покупець) та адміністратор. Клієнти взаємодіятимуть із сайтом магазину для перегляду і придбання товарів, а адміністратор буде управляти асортиментом і забезпечуватиме підтримку роботи магазину.

2.2.1 Функціональні вимоги для інформаційної системи

Функціональні вимоги – це вимоги, що визначають, що саме повинна робити система. Іншими словами, ці вимоги описують певні конкретні функції, можливості та операції, які система виконує для користувачів [13]. Саме функціональні вимоги задають та описують поведінку системи. Прикладом функціональних вимог є реєстрація користувачів, обробка даних, виконання розрахунків або надання певних інформаційних послуг.

Визначимо основні функціональні вимоги до системи для онлайн-реалізації товарів з особистим кабінетом користувача із забезпеченням типових можливостей електронної комерції для користувачів та адміністратора [14]:

- реєстрація та авторизація користувачів: система повинна дозволяти новим користувачам створювати облікові записи, а для зареєстрованих користувачів надає можливість входити до акаунту за своїм логіном та паролем;
- наявність особистого кабінету користувача: система повинна забезпечити наявність кабінету користувача, в якому клієнт може переглядати історію замовлень, редагувати власну інформацію та змінювати пароль. Також система повинна забезпечити можливість користувачу вийти із особистого кабінету;
- категорії товарів: система повинна дозволяти користувачу переглядати товар за основними категоріями в асортименті: сережки, намиста, кільця та браслети;
- товар: кожен товар повинен мати такі характеристики: назва, ціна, опис, характеристики;
- додавання та редагування товару (адміністратор): адміністратор повинен мати можливість додавати та редагувати товар без зміни програмного коду через адміністративну панель;
- акції (адміністратор): адміністратор повинен мати можливість уцінити товар, виставивши акційну ціну, яку буде бачити клієнт поруч із минулою ціною;
- каталог товарів: користувач повинен мати змогу переглядати каталог всіх товарів, а також за потреби переглядати товар за категоріями;
- кошик та оформлення замовлення: система повинна забезпечувати додавання обраних товарів до кошику, у разі потреби – видалення із кошику. Покупець повинен мати можливість оформити замовлення, вказавши дані для доставки та обравши доступний спосіб оплати;

- статус замовлення: у замовлення може бути 3 статуси – в обробці, в дорозі та завершено;
- управління замовленням (адміністратор): адміністратор повинен мати доступ до списку замовлень від клієнтів, бачити деталі кожного замовлення та змінювати статус замовлення.

2.2.2 Нефункціональні вимоги для інформаційної системи

Нефункціональні вимоги – це вимоги, що характеризують, як система виконує свої функції. Нефункціональні вимоги встановлюють критерії якості роботи системи, обмеження та характеристики, такі як швидкодія, зручність інтерфейсу, безпека тощо [13]. Саме нефункціональні вимоги завдають очікуваний рівень якості поведінки системи, які задають функціональні вимоги. Отже, сформуємо нефункціональні вимоги для онлайн-реалізації товарів біжутерії:

- зручність користування: система повинна бути інтуїтивно зрозумілою та мати зручний інтерфейс користувача. Клієнт повинен легко знаходити потрібні товари, швидко здійснювати покупку без зайвих кроків. Дизайн повинен бути приємним, відповідати тематиці біжутерії, містити анімацію, а кнопки й елементи навігації повинні бути чітко видимими. Процес реєстрації та оформлення замовлення повинні бути швидкими, простими та логічними;
- безпека даних: система повинна гарантувати захист конфіденційної інформації клієнтів. Особисті дані користувачів повинні бути збережені у зашифрованому вигляді, а доступ до адміністративної частини сайту повинен бути обмежений;
- сумісність і адаптивність: система повинна забезпечити можливість для користувачів користуватись сервісом у різних браузерах та на різних пристроях.

2.3 Формалізація основних задач інформаційної системи

Для успішного проєктування інформаційної системи одним із основних етапів є формалізація її основних задач. Формалізація задач ІС полягає в представленні основних функцій у формалізованому вигляді [15]. Інакше кажучи, кожна важлива задача записується чітко та однозначно, із вказанням вхідних даних, умов виконання та очікуваного результату. Даний підхід полегшує аналіз і подальшу реалізації системи, і є критично важливим етапом розробки, адже саме від ступеня формалізації задач значною мірою залежить функціонування всієї системи [16].

Основні задачі інформаційної системи виокремимо в таблиці 2.1 .

Таблиця 2.1 – Основні задачі інформаційної системи для онлайн-реалізації біжутерії з особистим кабінетом користувача

Назва задачі	Вхідні дані	Умови виконання	Очікуваний результат
Реєстрація користувача	E-mail, пароль	Неавторизований користувач натискає на кнопку профілю, натискає на посилання для реєстрації, користувача перенаправляє на сторінку реєстрації	Створено новий акаунт, користувача перенаправляє на головну сторінку, e-mail стає логіном
Авторизація користувача	Логін, пароль	Неавторизований користувач натискає на кнопку профілю, вводить дані і натискає кнопку для входу	Користувач увійшов до акаунту або повідомлення про помилку
Вихід з акаунту	Особистий кабінет користувача	Натискання кнопки «Вийти з акаунту»	Користувача перенаправляє на головну сторінку, користувач стає неавторизованим

Продовження таблиці 2.1

Назва задачі	Вхідні дані	Умови виконання	Очікуваний результат
Перегляд списку товарів	-	Користувач обирає категорію товарів	Відображається список товарів обраної категорії
Перегляд певного товару	Категорія товару або головна сторінка	Користувач при перегляді товару обирає будь-який товар	Користувача перенаправляє на сторінку товару
Додавання товару до кошика	Сторінка товару, кількість	Натискання кнопки «Додати до кошика»	Товар додається до кошика
Видалення товару із кошика	Кошик	Натиснути на кнопку видалення	Товар видалено із кошику
Оформлення замовлення	Кошик, контактні дані, спосіб доставки	Користувач знаходячись в кошику натискає «Далі», заповнює необхідні дані і натискає «Оформити замовлення»	Замовлення з’явилось в «Історія замовлень» в кабінеті користувача, присвоєно статус замовлення «В обробці»
Зміна статусу замовлення	Акаунт з правами адміністратора, ідентифікатор замовлення	Адміністратор обирає замовлення із списку, присвоює відповідний статус замовлення	Статус замовлення змінюється
Додавання нового товару	Акаунт з правами адміністратора, категорія товару, назва, ціна, опис, характеристика, зображення	Адміністратор натискає «Додати товар»	Товар з’являється в списку товарів
Постановка товару на уцінку	Акаунт з правами адміністратора, товар	Адміністратор вказує, що товар в акції, встановлює акційну ціну	Акція застосувалася до товару

Для візуалізації функціональних можливостей системи доцільно використати UML-діаграму прецедентів, яка дозволяє відобразити взаємодію користувача із основними функціями програмного забезпечення. На діаграмі прецедентів (див. рис. 2.2) зображено двох основних акторів – клієнт та

адміністратор, а також відповідні варіанти використання (use cases), які описують доступні їм функції. Для побудови логічної структури було використано зв’язки типу «include» – для обов’язкових підпроцесів та «extend» – для додаткових дій, які можуть виконуватись за певних умов. Така візуалізація допомагає чітко визначити, які дії має виконувати систему у відповідь на взаємодію з користувачем, і яким чином ці дії пов’язані між собою.

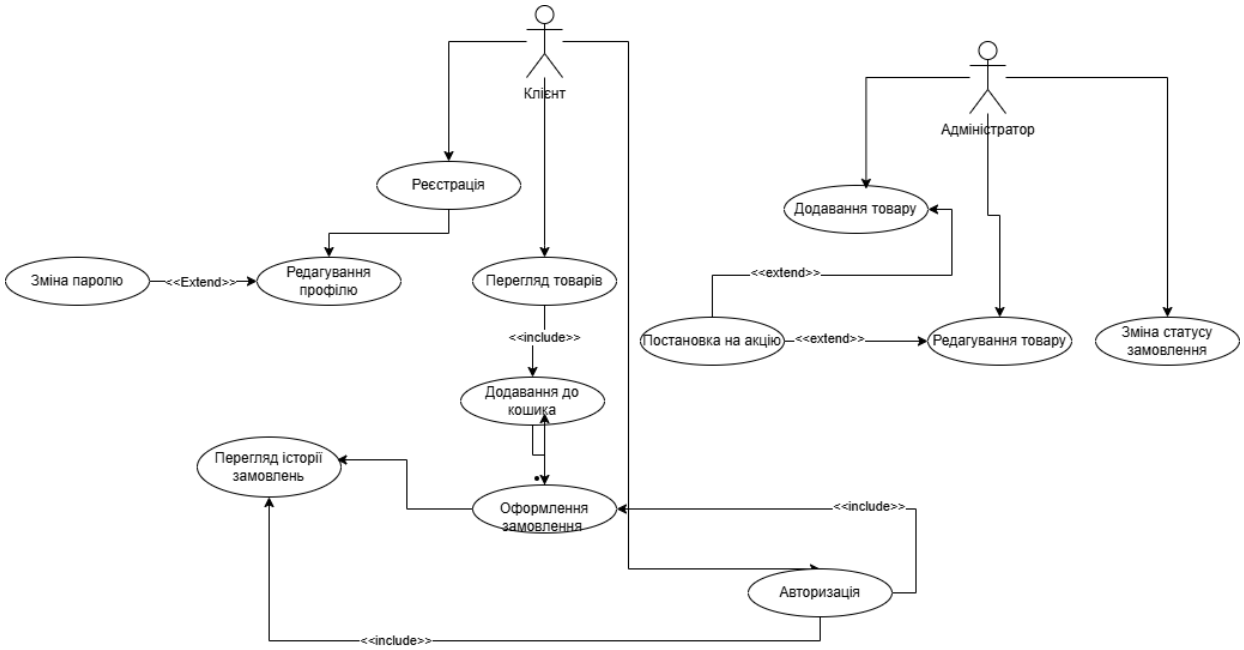


Рисунок 2.2 – UML-діаграма прецедентів взаємодії користувача з інформаційною системою

2.4 Проєктування архітектури інформаційної системи

Архітектура програмного забезпечення – це структура, яка описує, як система організована та те, як різні її частини взаємодіють між собою та із зовнішніми системами [17]. В залежності від підходу до побудови інформаційної системи, існує немало поширених архітектурних шаблонів. Щоб правильно підібрати архітектуру для проєкту, розглянемо декілька популярних шаблонів – моноліт, клієнт-сервер та модель-представлення-контролер (MVC).

2.4.1 Монолітна архітектура

Монолітна архітектура програмного забезпечення – це класичний підхід, при якому вся система реалізується як єдина цілісна програма з тісно пов’язаними між собою компонентами [18]. У випадку використання монолітної архітектури розроблювана система може бути у вигляді додатку, де інтерфейс користувача, бізнес-логіка та робота з даними об’єднані в одному виконуваному модулі. Основною перевагою монолітної архітектури програмного забезпечення є простота її реалізації та тестування [18], адже при дотриманні цього шаблону програмний код всього додатку є єдиним цілим, що полегшує загальне розуміння логіки розроблюваного застосунку. Схема застосування монолітної архітектури до інформаційної системи для онлайн-реалізації товарів з особистим кабінетом користувача зображена на рисунку 2.3.

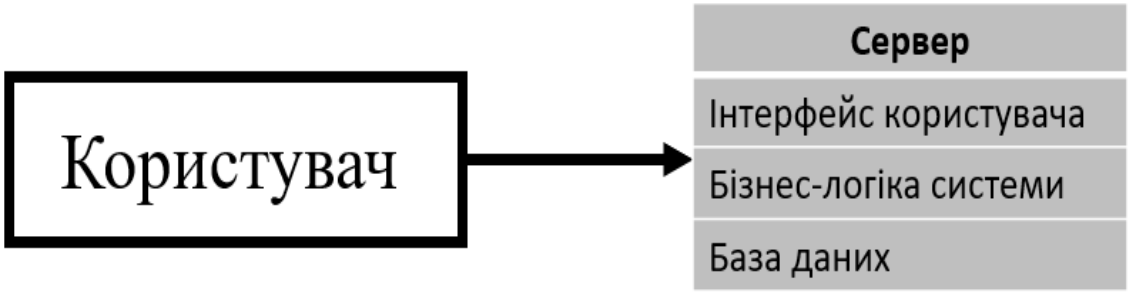


Рисунок 2.3 – Монолітна архітектура

Застосування монолітної архітектури є чудовим варіантом для розробки невеликих онлайн-магазинів, однак, у міру зростання магазину з’являються очевидні недоліки – масштабування і ускладнення внесення змін у програмний код великого застосунку [18]. Так, моноліт може «зекономити» час та ресурси при початковому проєктуванні та розробці системи, але може призвести до нераціонального використання ресурсів при спробі масштабувати застосунок через те, що програмний код є єдиним цілим, а отже і масштабувати систему

потрібно як єдине ціле, з відповідними змінами в усі частини коду, які відповідають як і за інтерфейс користувача, бізнес-логіку та роботу з даними.

2.4.2 Клієнт-серверна архітектура

Архітектура клієнт-сервер ґрунтується на розмежуванні системи на дві окремі частини: клієнтська частина, яка надсилає запити, і серверну, яка ці запити обробляє [18]. Централізоване виконання ключових функцій на сервері забезпечує ефективний розподіл навантаження і безпечне зберігання даних в єдиній базі [18]. При розробці інформаційної системи для онлайн-реалізації товарів з особистим кабінетом користувача та при дотриманні клієнт-серверної архітектури маємо такий розподіл ролей: в ролі клієнта виступає браузер, який надсилає запити через мережу, а віддалений сервер виконує всі потрібні обчислення та повертає результати назад до клієнта (див. рис. 2.4).

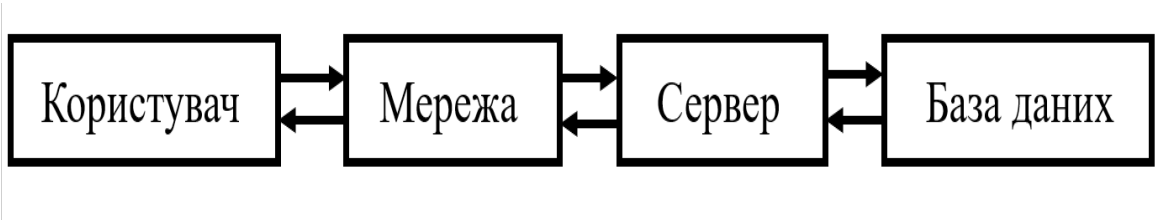


Рисунок 2.4 – Клієнт-серверна архітектура

В даній архітектурі вся бізнес-логіка та обробка запитів зосереджені на сервері, через що можна виявити вагомий недолік – у разі перевантаження сервера, збоїв або тимчасового відключення, клієнтська частина втрачає можливість виконувати навіть базові операції – наприклад, перегляд товарів чи додавання товару в кошик, тобто з’являється залежність працездатності системи від стабільності хостингу, інтернет-з’єднання та обчислювальних ресурсів серверної частини. Ці ризики є недоцільними для онлайн-магазину, адже повна недоступність системи призводить до збитків магазину та недоотриманої вигоди.

2.4.3 Архітектура MVC

Архітектура MVC (Модель-вид-контролер) є архітектурним шаблоном проєктування програмного забезпечення, що передбачає поділ системи на три основні компоненти: модель забезпечує логіку роботи з даними, вигляд забезпечує інтерфейс користувача, а контролер представляє собою модуль, що керує взаємодію між моделлю і виглядом [19]. Дана трирівнева структура дає змогу відокремити дані від представлення, таким чином зміни в інтерфейсі користувача мінімально впливають на роботу з даними, і навпаки – модифікації бізнес-логіки та структури даних не потребують переписування коду, що відповідає за інтерфейс користувача [19] (див. рис 2.5).

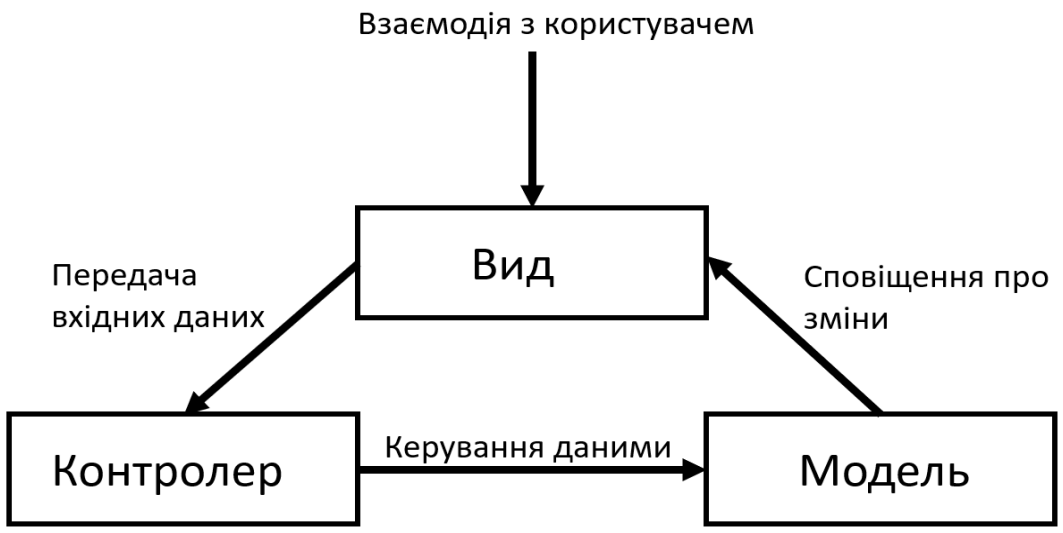


Рисунок 2.5 – Схема роботи архітектури MVC

Архітектура MVC є одним із найпоширеніших шаблоном у розробці сучасних веб-систем, адже його підтримують багато сучасних фреймворків (Laravel, ASP.NET MVC, Django). Також архітектура MVC особливо часто застосовується при створенні інтерфейсів веб-застосунків та мобільних застосунків та інтернет магазинів [20], адже при використанні цього шаблону кінцевий проєкт має чітко виражені рівні: рівень даних (товари, замовлення,

користувачі), рівень представлення (веб-сторінка) і рівень керування (обробка дій користувача, навігація застосунком, бізнес-логіка роботи застосунку).

Ще однією перевагою архітектурного шаблону MVC є те, що за рахунок поділу на модель, вигляд і контролер він має широкі можливості для масштабування і розширюваності системи. Завдяки такому поділу є можливість паралельно працювати над різними частинами застосунку – дизайнери та фронтенд розробники можуть зосередитись на створенні інтерфейсу, в той час як бекенд-розробники концентруються на логіці роботи з даними, при цьому зміни в одному компоненті системи (наприклад, оновлення дизайну веб-сторінок або модифікація структури бази даних) мінімально впливають на інші компоненти системи [19].

Враховуючи наведені аргументи, архітектурний шаблон модель-вигляд-контролер було обрано як найбільш оптимальну архітектуру для розробки інформаційної системи для онлайн-реалізації товарів з особистим кабінетом користувача, адже цей шаблон найбільше підходить функціональним та нефункціональним вимогам до системи через чітке розділення шарів системи та потенціалу для масштабування, який дозволяє залучити до розробки команду розробників для подальшого розширення функціоналу.

2.5 Розробка інформаційної моделі інформаційної системи

Одним із ключових етапів проєктування інформаційної системи є розробка її інформаційної моделі. Інформаційна модель інформаційної системи – це формалізоване подання знань про предметну область, тобто сукупність інформації, що характеризує істотні властивості та стани об’єктів чи процесів цієї області, а також їхні взаємозв’язки із зовнішнім середовищем [21].

Для розробки інформаційних систем інформаційну модель зазвичай представляють у вигляді концептуальної моделі даних – наприклад, у вигляді ER-діаграми (діаграми «сутність-зв’язок»). ER-модель відображає логічну

структуру даних системи і фактично виступає інформаційною моделлю, що відповідає реляційній моделі даних (тобто кожен тип об’єктів зберігається окремо, а між об’єктами встановлено визначені зв’язки) [22].

У контексті онлайн-магазину для реалізації біжутерії розроблена інформаційна модель відображає основні сутності та зв’язки, характерні для даної предметної області (електрона комерція з продажу прикрас). До ключових об’єктів інформаційної системи належать:

- користувач – зареєстрований учасник системи, що може переглядати асортимент і оформлювати замовлення; містить дані профілю, такі як: пошта, логін, пароль, ім’я, прізвище;
- товар – одиниця продукції, що пропонується в магазині; має такі характеристики як назва, опис, характеристики, ціна, акційна ціна (за наявності) та належить до певної категорії товарів;
- категорія – категорія або тип товарів (кільця, сережки, намиста, браслети), яка об’єднує групу схожих товарів;
- замовлення – заявка на покупку, оформлена користувачем; містить один або декілька обраних клієнтом товарів, дату та час замовлення та інформацію про покупця (користувача, що здійснив замовлення);
- статус замовлення – показник поточного етапу обробки замовлення (в обробці, в дорозі, завершено).

Зазначені об’єкти мають певні атрибути і пов’язані між собою логічними зв’язками (див. табл. 2.2) .

Для більш детального зображення інформаційної моделі інформаційної системи прийнято використовувати ER-діаграми. Схему інформаційної моделі онлайн-магазину біжутерії, яка відображає всі перелічені сутності та зв’язки між ними у вигляді ER-діаграми зображено на рисунку 2.6.

Таблиця 2.2 Атрибути об’єктів та логічні зв’язки між ними

Назва об’єкту	Атрибути	Зв’язки між об’єктами
Користувач	Id, логін, прізвище, ім’я, email, пароль	Має зв’язок 1:N із «Замовленням» (один користувач може мати багато замовлень)
Товар	Id, назва, опис, характеристика, ціна, акційна ціна (опціонально), категорія_id	Має зв’язок N:1 із «Категорією» (кожен товар належить до однієї категорії) та зв’язок N:M із «Замовленням» через таблицю «Товар у замовленні»
Категорія	Id, назва	Має зв’язок 1:N із «Товаром» (одна категорія включає багато товарів).
Замовлення	Id, користувач_id, дата, статус_ id, адреса відправлення	Має зв’язок N:1 із «Користувачем». Має зв’язок N:1 із «Статусом замовлення». Має зв’язок N:1 із «Товаром» через таблицю «Товар у замовленні».
Статус замовлення	Id, назва статусу	Має зв’язок 1:N із «Замовленням» (один статус може повторюватись у багатьох замовленнях)
Товар у замовленні	Id, замовлення_id, товар_id, кількість	Є проміжною таблицею для зв’язку M:N між «Товаром» і «Замовленням». Має зв’язок N:1 із «Товаром» та N:1 із «Замовленням».

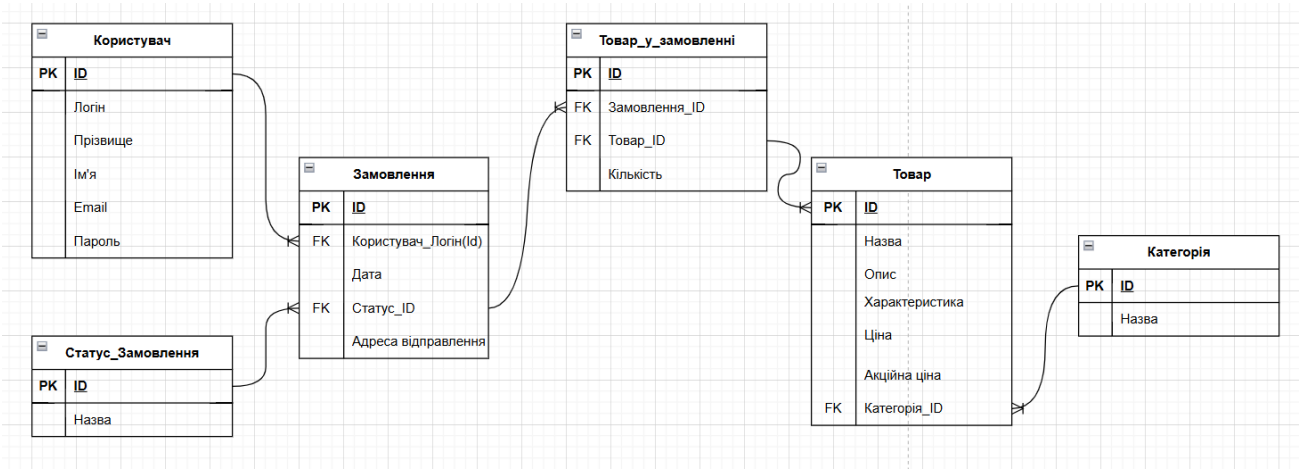


Рисунок 2.6 – Інформаційна модель інформаційної системи у вигляді ER-діаграми

2.6 Розробка структури інформаційної системи

Структура сайту інтернет-магазину – це схема блоків, збудованих у певній логічній послідовності, яка візуалізує всі сторінки онлайн-магазину та показує їх ієрархію і взаємозв'язки [23]. На етапі проєктування інформаційної системи інтернет-магазину особливу увагу приділяється розробці його структури, аби забезпечити зручність користування та логічну організацію інформації для клієнтів, адже якісно спроєктована структура допоможе користувачам швидку знаходити потрібні товари і швидко здійснювати покупки, не витрачаючи зайвого часу на пошук. До структури інтернет-магазину повинні входити такі сторінки, як головна сторінка, сторінка каталогу товару, картка товару, кошик, особистий кабінет, сторінка оформлення замовлення та службові сторінки.

Перше, з чим зустрічається користувач – це головна сторінка сайту. У нашому проєкті головна сторінка виконує роль вітрини і повинна містити декілька важливих блоків. По-перше, це верхня частина сайту (шапка) з елементами навігації: логотипом і назвами категорій товару, а також з кнопками «Профіль» та «Кошик» для швидкого доступу до особистого кабінету користувача та кошику покупок. По-друге, центральна область головної

сторінки, яка має на меті привернути увагу користувача до певної категорії товарів, отже влучним рішенням буде помістити банер – великий візуальний блок із пропозицією, по якому клієнт може перейти. По-третє, це нижня частина сайту (футер), у якому передбачено набір службових інформаційних сторінок і корисних посилань. Футер слугуватиме «довідковим розділом», де користувач може швидко знайти допоміжну інформацію, не відволікаючись від процесу вибору товарів. Зокрема, планується розмістити такі сторінки:

- догляд за прикрасами: рекомендації щодо правильного догляду за біжутерією;
- доставка та повернення: інформація про способи та терміни доставки замовлень, вартість транспортування, а також умови повернення чи обміну товарів; ця сторінка повинна забезпечити прозорість умов покупки для клієнтів;
- часті питання: відповіді на типові запитання покупців;
- наша історія: сторінка про компанію – історія створення бренду, місія, цінності, досягнення;
- паспорт товару – розділ з додатковими відомостями про якість продукції;
- соціальні мережі: посилання на офіційні сторінки магазину в соцмережах для залучення аудиторії та підтримання зворотного зв’язку з клієнтами.

З урахуванням описаних функціональних компонентів та навігаційної логіки було сформовано загальну структуру майбутнього сайту інтернет-магазину у вигляді схеми, що відображає логічну ієрархію основних сторінок сайту (див. рис. 2.7).

Розроблена структура сайту підтримує зручну поведінку користувача починаючи від першого відвідування до оформлення замовлення та подальшої роботи з особистим кабінетом.

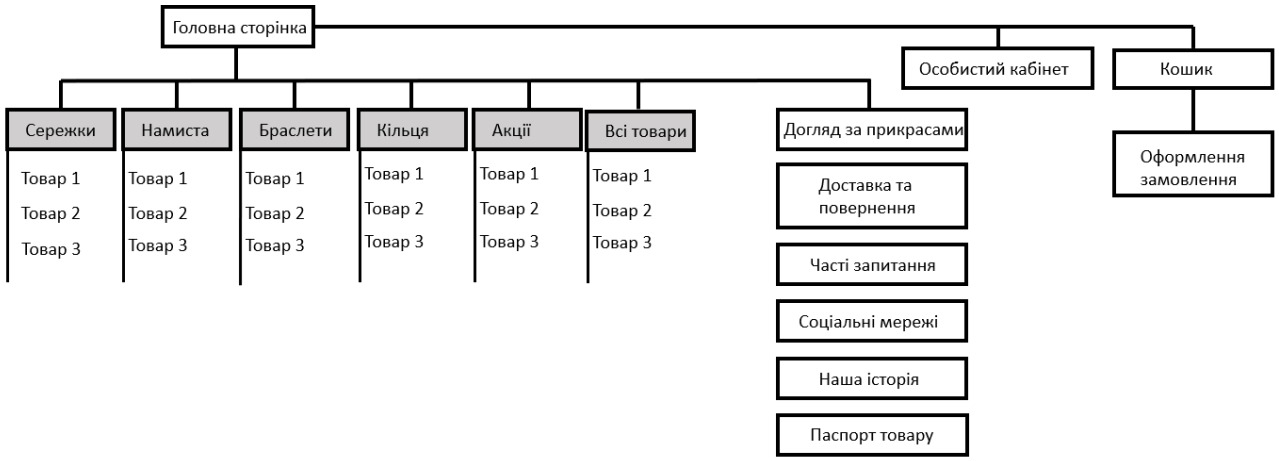


Рисунок 2.7 – Структурна схема сайту інтернет-магазину біжутерії

Основні кроки взаємодії користувача із системою виглядають наступним чином:

- перехід до категорії товарів: клієнт заходить на головну сторінку і обирає один із розділів каталогу, після чого переходить на сторінку обраної категорії;
- перегляд списку товарів: на сторінці категорії відображається список доступних товарів із назвою товару та ціною часті питання: відповіді на типові запитання покупців;
- відкриття сторінки товару: зацікавившись конкретним товаром, користувач натискає на нього і переходить на окрему сторінку товару (картку товару). На цій сторінці міститься детальна інформація: фото високої якості, розгорнутий опис прикраси, характеристики, ціна;
- додавання товару до кошика: ознайомившись з деталями, користувач додає обраний товар до кошика, натискаючи кнопку «Додати до кошика» на сторінці товару. Система оновлює стан кошика, а користувачу надсилається відповідне повідомлення;
- оформлення замовлення: коли всі бажані товари зібрано в кошику, користувач переходить до його перегляду. На сторінці кошика він перевіряє список обраних позицій, їх кількість, за потреби може змінити кількість або видалити товар з кошику. Користувач бачить підсумкову вартість, після чого натискає кнопку «Оформити

замовлення». Далі система перенаправляє на сторінку оформлення, де покупець заповнює всі необхідні дані для доставки та оплати і підтверджує замовлення. В результаті замовлення зберігається в системі зі статусом «в обробці», а користувач отримує підтвердження на екрані, а саме замовлення він може побачити в «історії замовлень» в особистому кабінеті;

- вхід та реєстрація: перед здійсненням покупки користувачу потрібно увійти в свій профіль. Авторизація здійснюється через введення логіну та паролю на сторінці «Профіль», після чого покупець потрапляє до особистого кабінету. Якщо клієнт новий, він може попередньо зареєструватися, заповнивши реєстраційну форму;
- перегляд історії замовлень: в особистому кабінеті користувач має можливість переглянути історію своїх замовлень;
- редагування особистих даних: крім перегляду замовлень, особистий кабінет надає можливість керувати своїми даними. Користувач може відредагувати персональну інформацію: змінити ім’я, прізвище або пароль.

Оскільки адміністративна частина сайту не доступна для звичайного користувача, її структура на схемі не зображена, так як адміністратор взаємодіє із системою через окрему панель керування (бек-енд частини сайту). Основні функції адміністратора включають:

- керування замовленнями: перегляд усіх замовлень, що надходять від покупців та зміна їх статусу в процесі обробки;
- управління товарами: додавання нових позицій на сайт з усіма характеристиками, редагування характеристик уже наявних товарів або видалення товарів, що більше не продаються;
- керування акціями: створення та налаштування акційних пропозицій.

Отже, усі перелічені кроки реалізуються послідовно та логічно завдяки продуманій структурі сайту. Інтуїтивна навігація та зрозумілий інтерфейс гарантують позитивний користувацький досвід, а завдяки адміністративній панелі власник або модератор сайту має повний контроль над вмістом і

процесами інтернет магазину, а отже може постійно оновлювати інформацію на сайті, стежити за актуальністю цін та додавати нові продукти, а також своєчасно реагувати на дії користувачів (обробка замовлень).

2.7 Висновки до 2 розділу

В цьому розділі кваліфікаційної роботи було здійснено комплексне проєктування інформаційної системи для онлайн-реалізації біжутерії з особистим кабінетом користувача. В результаті проведеного аналізу предметної області було обґрунтовано доцільність вибору такої категорія товару, як біжутерія, яка характеризується високою привабливістю для інтернет-покупця, мобільністю логістики та низьким рівнем офлайн-конкуренції.

На основі аналізу було сформульовано функціональні та нефункціональні вимоги до майбутньої системи, що дозволило окреслити основні задачі, які повинна реалізовувати система. Ці задачі були формалізовані у вигляді таблиці із зазначенням вхідних даних, умов виконання та очікуваних результатів.

Було проаналізовано типові архітектурні шаблони для побудови інформаційних систем, серед яких найбільш відповідною визнано архітектуру MVC, що забезпечує логічне розділення рівнів даних, логіки та представлення. Розроблено інформаційну модель системи у вигляді ER-діаграми, яка відображає основні об’єкти, їх атрибути та взаємозв’язки між ними.

Також сформовано структурну схему сайту інтернет-магазину, яка демонструє ієрархію сторінок, логіку переходів та взаємодію користувача з системою. Окремо розглянуто функціональні особливості адміністративної частини, яка не представлена на клієнтській схемі, проте є важливим аспектом для управління контентом та процесами.

Таким чином, у цьому розділі закладено методологічну та логічну основу для подальшої реалізації інформаційної системи, що відповідає сучасним вимогам до онлайн-торгівлі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОНЛАЙН РЕАЛІЗАЦІЇ ТОВАРІВ З ОСОБИСТИМ КАБІНЕТОМ КОРИСТУВАЧА

3.1 Вибір засобів реалізації інформаційної системи

У рамках кваліфікаційної роботи для реалізації інформаційної системи обрано стек технологій Python/Django для розробки бекенд-частини сайту та HTML/CSS/Bootstrap для розробки фронтенд частини сайту, з використанням СУБД SQLite. Мова Python є популярною високорівневою мовою загального призначення з підтримкою великої кількості бібліотек та модулів для веб-розробки [24][25]. Вона широко використовується в індустрії, на ній розроблено багато всесвітньовідомих сайтів, таких як Google, Netflix, та підтримується активною спільнотою, що зменшує витрати на розробку завдяки повторному використанню існуючого коду. Використання фреймворку Django дозволяє значно пришвидшити створення веб-додатку завдяки концепції «batteries included»: фреймворк надає вбудовані механізми роботи з URL, ORM (модельно-орієнтована абстракція БД), аутентифікації користувачів, захисту від поширених атак, таких як CSRF або XSS, а також готову адміністративну панель [24][25]. Django є одним з найпопулярніших Python-фреймворків, за рахунок чого існує велика кількість документації та розширень; його застосовують як для API, так і для класичних веб-додатків, кількість яких оцінюється тисячами [24]. HTML та CSS забезпечують базову розмітку інтерфейсу, а Bootstrap дозволяє швидко реалізувати адаптивний та привабливий дизайн без необхідності розробки стилів.

Як систему керування базами даних обрано SQLite – легка вбудована СУБД без клієнт-серверної архітектури. Вона не вимагає окремого налаштування чи підтримки сервера: вся база зберігається в одному файлі на диску. Завдяки відсутності серверу та малій вазі SQLite дуже просто інтегрувати

в додаток і реалізувати подальшу підтримку [26]. Ця СУБД ідеально підходить для невеликих або середніх проєктів, де кількість одночасних підключень обмежена, а пріоритетом є швидке прототипування та простота розгортання.

Водночас розглядались альтернативні технології. Наприклад, замість Python можна було б використати мову програмування PHP з фреймворком Laravel. PHP поширений у веб-розробці та добре інтегрується з популярними CMS, проте Python надає більш чистий синтаксис і потужні інструменти для побудови складних систем [27]. Laravel пропонує розробникам гнучку MVC-архітектуру і багатий набір готових пакетів, але має крутіший поріг входження в порівнянні з Django [28]. Для фронтенду альтернативами могли би бути React або інші JavaScript-фреймворки, які забезпечують динамічні SPA, однак це ускладнило б проєкт і вимагає більше часу на розробку простого інтерфейсу, знижуючи ефективність використання ресурсів проєкту. Як альтернативи до СУБД SQLite було розглянуто MySQL та PostgreSQL – потужні клієнт-серверні СУБЖ з великим набором можливостей для високонавантажених проєктів, але потребують окремого серверного середовища і складнішого адміністрування, що також знижує ефективність використання ресурсів проєкту.

Таким чином, перед вибором конкретних технологій для реалізації інформаційної системи було доцільно провести порівняльний аналіз можливих альтернатив з урахуванням специфіки проєкту, вимог до функціональності, простоти розгортання та підтримки. Зведене порівняння основних характеристик використаних та альтернативних рішень наведено в таблиці 3.1.

Обраний стек технологій вирізняється простотою розгортання, високою ефективністю використання ресурсів та достатньою продуктивністю для вирішення поставлених завдань та вимог до інформаційної системи. Python/Django дозволяють швидко реалізувати бізнес-логіку і надають готові засоби для роботи з базами даних та безпеки [28], HTML/CSS/Bootstrap підвищують ефективність розробки інтерфейсу користувача, а SQLite мінімізує накладні витрати на інфраструктуру. У сукупності ці технології забезпечують

баланс між зручністю розробки, швидкістю виконання та відповідністю вимогам системи.

Таблиця 3.1 Порівняння основних характеристик альтернативних технологій

Категорія	Використана технологія	Альтернативна технологія	Основні характеристики
Мова програмування	Python	PHP	Python має чистий синтаксис і розвинену екосистему, яка дозволяє створювати безпечні та масштабовані веб-додатки [27]; PHP широко використовується при веб-розробці, легко інтегрується із CMS, але поступається Python зручністю синтаксису.
Веб-фреймворк	Django	Laravel	Django має вбудовані ORM та систему аутентифікації, простий у вивченні, через що відзначається високою продуктивністю [28]; Laravel надає багату функціональність, проте складніший у налаштуванні і розгортанні і менш безпечний при стандартних налаштуваннях.

Продовження таблиці 3.1

СУБД	SQLite	MySQL	SQLite – вбудована безсерверна СУБД, легко налаштовується і має малий розмір, що спрощує розгортання [26]; MySQL – потужна клієнт-серверна СУБД з розширеними функціями (масштабованість, складні типи даних), але потребує складнішого адміністрування і більшого використання ресурсів проєкту.

3.2 Структура програмного проєкту

У Django використовується модифікована архітектура MVC, яка має назву MTV(Model-Template-View). Вона зберігає ключові принципи MVC, однак змінює назви ролей для адаптації до особливостей веб-розробки на Python. Model – відповідає за зберігання й доступ до даних, View обробляє HTTP-запити, а Template формує кінцеве представлення (HTML) для користувача [29, с. 8-9]. Проєкт у Django – це контейнер із набором конфігурацій і додатків, що разом формують веб-сайт. Кожен додаток виконує окрему функцію, наприклад аутентифікацію користувачів або управління каталогом товарів. Така модульна структура сприяє повторному використанню коду й легшій підтримці проєкту. Додатки мають власні файли models.py, views.py, urls.py, templates/ та static/, що дозволяє чітко розділити відповідальність компонентів.

В проєкті наявні такі основні модулі:

– accounts: модуль для зберігання інформації про профілі користувачів. У models.py міститься профіль користувача, його данні та передбачена можливість для оновлення інформації:

```
from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
class RegisterForm(UserCreationForm):
    email = forms.EmailField(required=True)
    first_name = forms.CharField(required=False)
    last_name = forms.CharField(required=False)
    class Meta:
        model = User
        fields = ['username', 'email', 'first_name', 'last_name', 'password1', 'password2']
class UserUpdateForm(forms.ModelForm):
    class Meta:
        model = User
        fields = ['first_name', 'last_name', 'email']
```

Файл admin .py реєструє модель для адміністратора, а views.py визначає представлення, такі як редагування профілю. Файл urls.py визначає маршрути для цього модулю. Шаблони templates/accounts/register.html та templates/accounts/profile.html відповідають за реєстрацію, вхід в акаунт та відображення особистого кабінету користувача;

– userauth: модуль для входу в акаунт, в якому для цього використовується стандартний інструмент django.contrib.auth. ;
– products: модуль каталогу товарів. У models.py описуються сутності категорій товарів та самих товарів:

```
from django.db import models
class Category(models.Model):
```

```
    name = models.CharField(max_length=100)
    image = models.ImageField(upload_to='category_images/', blank=True,
     null=True)
    def __str__(self):
      return self.name
class Product(models.Model):
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    name = models.CharField(max_length=200)
    description = models.TextField()
    characteristics = models.TextField(blank=True, null=True)
    price = models.DecimalField(max_digits=8, decimal_places=2)
    discount_price = models.DecimalField(max_digits=8, decimal_places=2,
     blank=True, null=True)
    is_discounted = models.BooleanField(default=False)
    def get_price(self):
      return self.discount_price if self.is_discounted and self.discount_price else
     self.price
    def __str__(self):
      return self.name
class ProductImage(models.Model):
    product = models.ForeignKey(Product, on_delete=models.CASCADE,
     related_name='images')
    image = models.ImageField(upload_to='product_images/')
    def __str__(self):
      return f"Image for {self.product.name}"
```

У views.py знаходяться класи для списку та детального перегляду товарів, а в urls.py – відповідні маршрути. Шаблони в templates/products/ відображають сторінку товару, каталог акційних товарів, всіх товарів та товарів за категорією;

– cart: модуль кошика. У ньому зберігаються вибрані користувачем товари до оформлення замовлення. У models.py представлено модель CartItem:

```
from django.db import models
from django.conf import settings
from products.models import Product
class CartItem(models.Model):
user = models.ForeignKey(settings.AUTH_USER_MODEL,
on_delete=models.CASCADE, null=True, blank=True)
session_key = models.CharField(max_length=40, null=True, blank=True)
product = models.ForeignKey(Product, on_delete=models.CASCADE)
quantity = models.PositiveIntegerField(default=1)
added_at = models.DateTimeField(auto_now_add=True
```

У views.py – обробники додавання та видалення товарів із кошика, оновлення ціни, зміна кількості товару, а шаблон templates/cart показує поточний вміст кошика;

– orders: модуль оформлення замовлення та історії замовлення. Models.py містить моделі Order та OrderItem:

```
class Order(models.Model):
user = models.ForeignKey(User, on_delete=models.SET_NULL, null=True,
blank=True, verbose_name="Користувач")
STATUS_CHOICES = [
('processing', 'В обробці'),
('shipped', 'В дорозі'),
('completed', 'Завершено'),
]
full_name = models.CharField("ПІБ", max_length=100)
phone = models.CharField("Номер телефону", max_length=20)
region = models.CharField("Область", max_length=100)
```

```
    city = models.CharField("Місто", max_length=100)
    post_office = models.CharField("Відділення Нової Пошти",
      max_length=100)
    created_at = models.DateTimeField(auto_now_add=True)
    status = models.CharField("Статус", max_length=20,
      choices=STATUS_CHOICES, default='processing')
    def __str__(self):
      return f"Замовлення #{self.id} – {self.full_name}"
    def get_total_sum(self):
      return sum(item.get_total_price() for item in self.items.all())
class OrderItem(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE,
      related_name='items')
    product = models.ForeignKey(Product, on_delete=models.SET_NULL,
      null=True)
    quantity = models.PositiveIntegerField()
    def __str__(self):
      return f"{self.product.name} x {self.quantity}"
    def get_total_price(self):
      return self.product.get_price() * self.quantity
```

У views.py реалізуються процедури оформлення замовлення та збереження замовлення, а templates/orders/ – відповідні сторінки оформлення замовлення;

– backend: ядро проєкту, створене за допомогою команди Django-admin-startproject. У ньому зосереджені глобальні налаштування, що регулюють роботу всієї системи. Файл settings.py містить конфігурацію бази даних, списки встановлених додатків, шляхи до шаблонів та статичних файлів, а також параметри безпеки, файл urls.py відповідає за маршрутизацію, підключаючи всі додатки та формуючи єдину структуру. Крім цього, саме у файлі settings.py підключається власний

контекстний процесор cart.context_processors.cart_item_count, що дозволяє відображати кількість товарів у кошику в усіх шаблонах. Завдяки параметру APP_DIRS = True, Django автоматично шукає шаблони у відповідних каталогах templates/ для кожного додатку. Також в urls.py визначено маршрут на головну сторінку index.html, яка є вітальною сторінкою інтерфейсу сайту.

3.2.1 Конфігурація проєкту

Ключові файли на рівні проєкту формують загальну конфігурацію проєкту. Файл settings.py містить усі налаштування проєкту, такі як підключення до бази даних, безпека, список встановлених додатків INSTALLED_APPS, а urls.py описує маршрутизацію – зіставляє URL-адреси з відповідними представленнями [30]. В settings.py вказано підключення додатків: INSTALLED_APPS = [

```
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'userauth',
    'products',
    'cart',
    'orders',
    'widget_tweaks',
]
```

В urls.py виконується підключення маршрутів всіх додатків:

```
from django.contrib import admin
```

```
from django.urls import path, include
from django.views.generic import TemplateView
from django.conf import settings
from django.conf.urls.static import static
from products import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', TemplateView.as_view(template_name='index.html'), name='home'),
    path('accounts/', include('accounts.urls')),
    path('', include('django.contrib.auth.urls')),
    path('product/', include('products.urls')),
    path('', views.home, name='home'),
    path('cart/', include('cart.urls')),
    path('orders/', include('orders.urls'))
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Таким чином, settings.py та urls.py разом задають глобальну конфігурацію та «зміст» (table of contents) проєкту [30].

3.2.2 Шаблони та статичні файли

HTML-шаблони зазвичай розміщуються у папках templates/ кожного додатку або у загальній директорії templates/, визначеній у налаштуваннях. За умовчанням у settings.py параметр TEMPLATES містить APP_DIRS: True, тобто Django автоматично шукає шаблони в теці templates кожного встановленого додатку. Статичні файли, такі як зображення або CSS, зберігаються в теці static/ відповідного додатку або в глобальних теках, зазначених в налаштуваннях через STATICFILES_DIRS. Для цього необхідно додати в settings.py:

```
STATIC_URL = '/static/'
```

STATICFILES_DIRS = [BASE_DIR / 'static']

Під час розробки вбудований сервер Django автоматично обслуговує статичні файли (при DEBUG=True), а в продакшені їх зазвичай збирають і сервують окремо. Така організація дозволяє підтримувати шаблони і статичні ресурси проєкту у впорядкованому вигляді, зі зрозумілою прив’язкою до кожного модуля.

3.2.3 Робота з базою даних

У проєкті використовується вбудована база даних SQLite, яка підтримується Django за замовчуванням. Операції з базою виконуються за допомогою ORM (Object-Relational Mapping), яка дозволяє взаємодіяти з таблицями через Python-класи, не прописуючи SQL-запити вручну, тобто ORM автоматично генерує SQL-запити. Наведемо декілька прикладів SQL, які відповідають типовим викликам у коді:

- отримати всі товари певної категорії: код Product.objects.filter(category__name='Сережки') відповідає запиту
  SELECT * FROM products_product
  = products_category.id
  WHERE products_category.name = 'Сережки' ;
- отримати всі товари в кошику користувача: CartItem.objects.filter(user=request.user) відповідає запиту
  SELECT * FROM cart_cartitem
  WHERE user_id = <id користувача>;
- додати новий товар до таблиці Product:
  INSERT INTO products_product
  (name, description, price, category_id, is_discounted)
  VALUES ('Кільце Heart', 'Срібне кільце у формі серця', 450.00, 2, FALSE);

- отримати всі акційні товари: Product.objects.filter(is_discounted=True) відповідає SELECT * FROM products_product
  WHERE is_discounted = 1;
- отримати зображення певного товару: product.images.all() відповідає запиту SELECT * FROM products_productimage
  WHERE product_id = <id товару>; .

Отже, використання ORM у Django забезпечує зручність та швидкість розробки, дозволяючи абстрагуватися від рутинного SQL-коду. Тим не менш, у разі потреби ORM дозволяє переглянути та оптимізувати скомпільовані запити, що особливо важливо при роботі з великим обсягом даних для оптимізації продуктивності.

3.3 Реалізація функціоналу користувача

Однією із ключових складових інформаційної системи для онлайн-торгівлі є модуль взаємодії з користувачем. Наявність особистого кабінету, можливість авторизації, реєстрації та безпечного доступу до персональних даних створюють позитивний клієнтський досвід та підвищують рівень лояльності користувача до платформи та самого магазину. Розглянемо реалізацію базового функціоналу користувача: процес реєстрації та входу в систему, структуру особистого кабінету а також механізми обмеження доступу до профілю. Реалізація здійснена з використанням стандартних інструментів Django, які забезпечують відповідність системи нормам сучасних вимог безпеки та її ефективність.

3.3.1 Реєстрація та авторизація користувача

Для зручності користувача в інформаційній системі реалізовано простий та інтуїтивно зрозумілий механізм переходу до особистого кабінету. Загальна

логіка дій, починаючи з будь-якої сторінки сайту, зображена на блок-схемі (рис. 3.1).

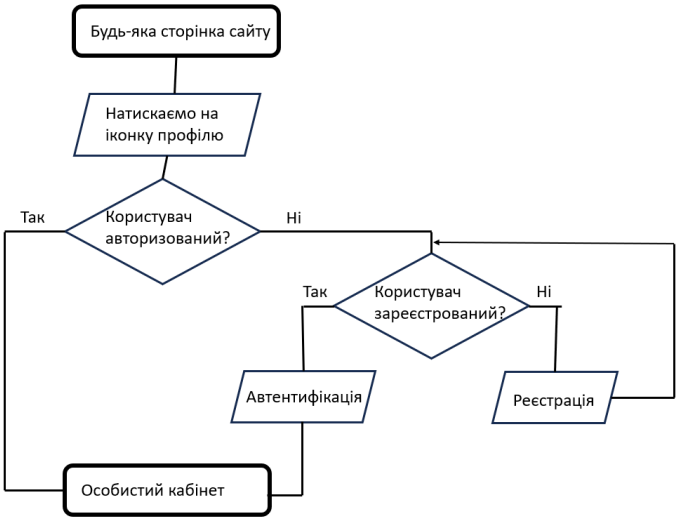


Рисунок 3.1 – Логіка авторизації та реєстрації користувача

Користувач, натискаючи на іконку профілю, потрапляє або до сторінки авторизації/реєстрації, або безпосередньо до особистого кабінету у випадку, якщо вже авторизований. Django має вбудовану систему автентифікації, що надає модель користувача, стандартні представлення (view) для входу і виходу, а також готові форми для створення нових облікових записів [31]. Завдяки цьому можна швидко реалізувати базові операції реєстрації та авторизації без написання низькорівневого коду роботи з сесіями чи хешування паролів. У даному проєкті процес реєстрації реалізовано через власну форму RegisterForm, що успадковує клас UserCreationForm. Використання кастомної форми на основі стандартної дозволяє за потреби змінити або доповнити поля, які вводить користувач під час реєстрації, а також накласти додаткові правила валідації. Після заповнення форми та її валідації створюється новий об’єкт користувача шляхом виклику методу save(). Форма реєстрації відображається у шаблоні register.html (див.рис. 3.2) та містить поля для введення необхідних даних нового користувача (логін, пошта, прізвище, ім’я, пароль двічі):

```
from django import forms
```

```
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
class RegisterForm(UserCreationForm):
    email = forms.EmailField(required=True)
    first_name = forms.CharField(required=False)
    last_name = forms.CharField(required=False)
    class Meta:
        model = User
                fields = ['username', 'email', 'first_name', 'last_name', 'password1', 'password2']
```

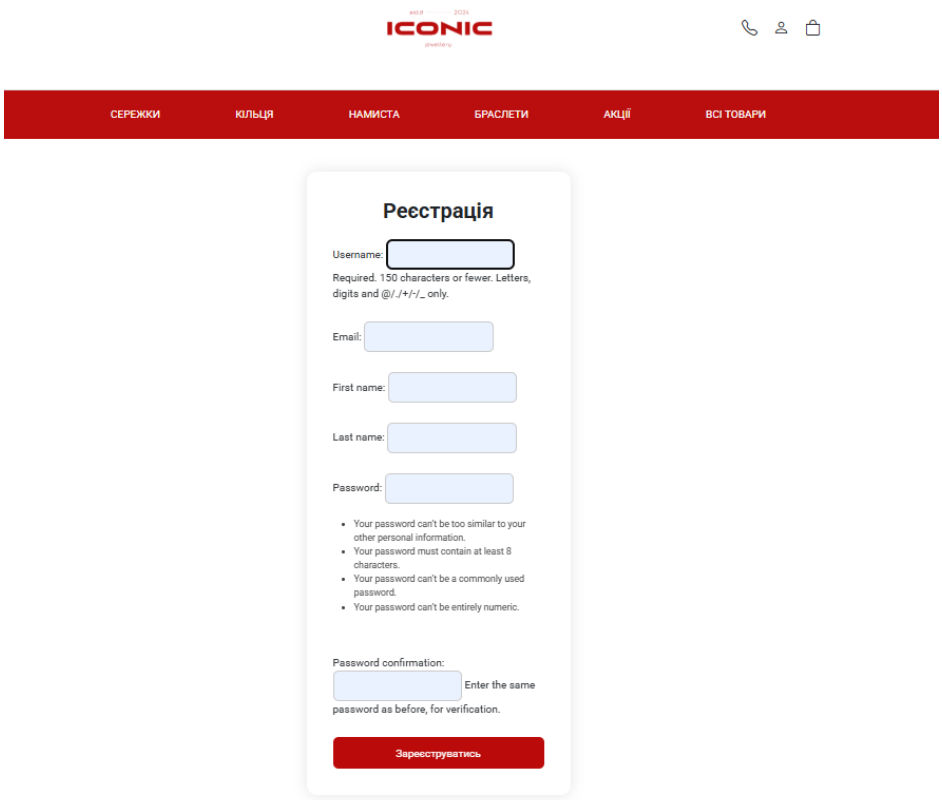


Рисунок 3.2 – Форма реєстрації

Після успішної реєстрації користувач може увійти до системи за допомогою свого логіна і пароля через стандартну сторінку входу , яку реалізовано в Django засобами додатку Django.contib.auth , яка відображається у шаблоні login.html (див. рис 3.3).

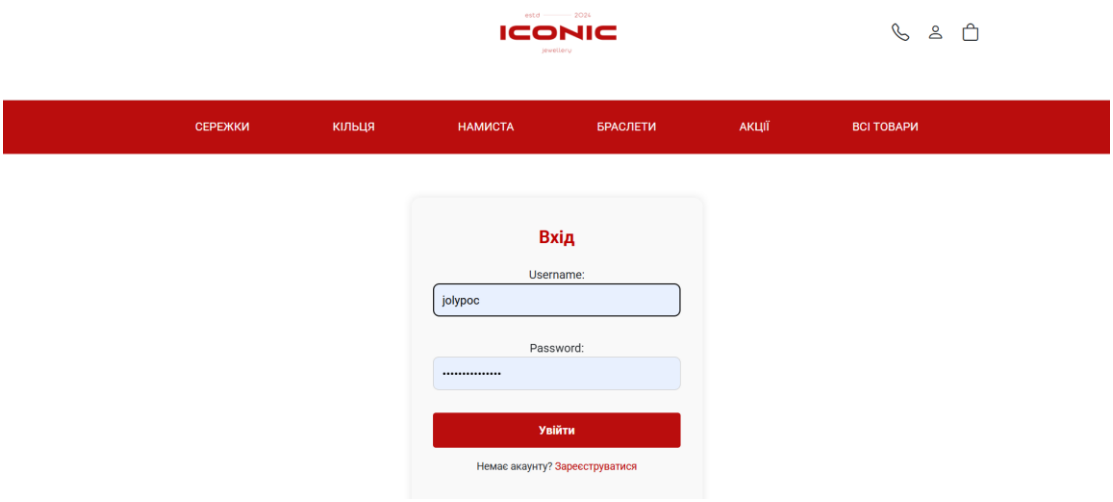


Рисунок 3.3 – Авторизація користувача

Авторизація (вхід користувача) здійснюється через стандартне представлення Django – достатньо налаштувати URL на LoginView або використати функцію login з пакету автентифікації. Цей механізм перевіряє введені користувачем облікові дані та створює сесію при успішному вході.

3.3.2 Особистий кабінет користувача

Для зареєстрованого й автентифікованого користувача в системі реалізовано сторінку особистого кабінету, в якій користувач може потрапити після входу в акаунт або натиснувши на кнопку профілю з будь-якої сторінки сайту. В особистому кабінеті користувач може переглянути історію зроблених ним замовлень. Даний функціонал реалізовано у вигляді представлення profile_view, яке відображає сторінку профілю.

Оновлення персональних даних здійснюється через форму UserUpdateForm, що дозволяє редагувати поля моделі користувача (такі як ім’я, прізвище, email). Ця форма заповнюється поточними даними користувача (параметр instance=request.user), відображається в шаблоні profile.html (див. рис. 3.4) і дозволяє відправити зміни на сервер.

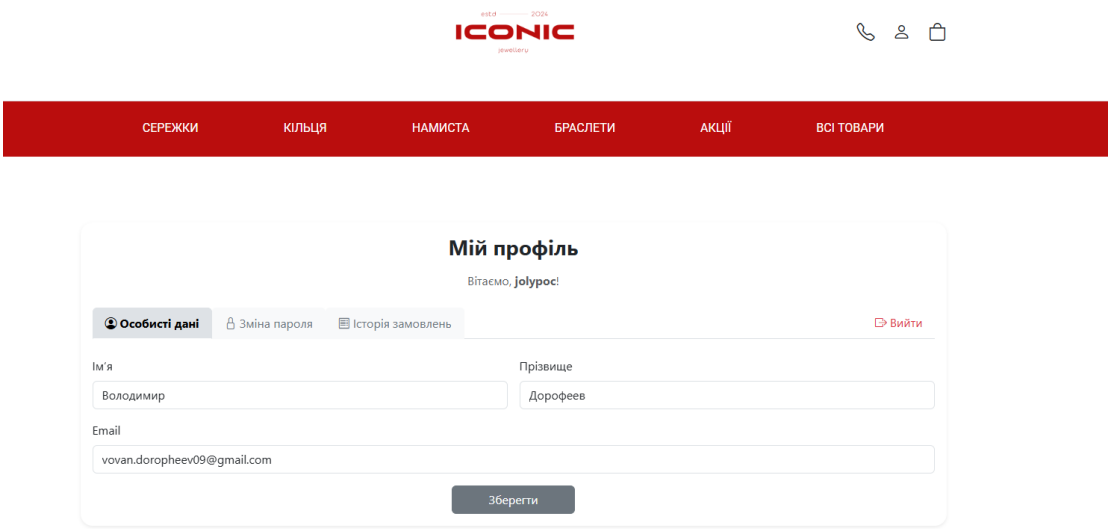


Рисунок 3.4 – Особистий кабінет користувача

На боці сервера profile_view обробляє запит: у випадку метода POST відбувається заповнення форми переданими даними та перевірка через form.is_valid() . Якщо нові дані пройшли валідацію, вони зберігаються у базі методом save(), після чого форма відображається вже з оновленими значеннями.

Крім оновлення основних даних, на сторінці особистого кабінету передбачено можливість зміни пароля користувача (див. рис 3.5).

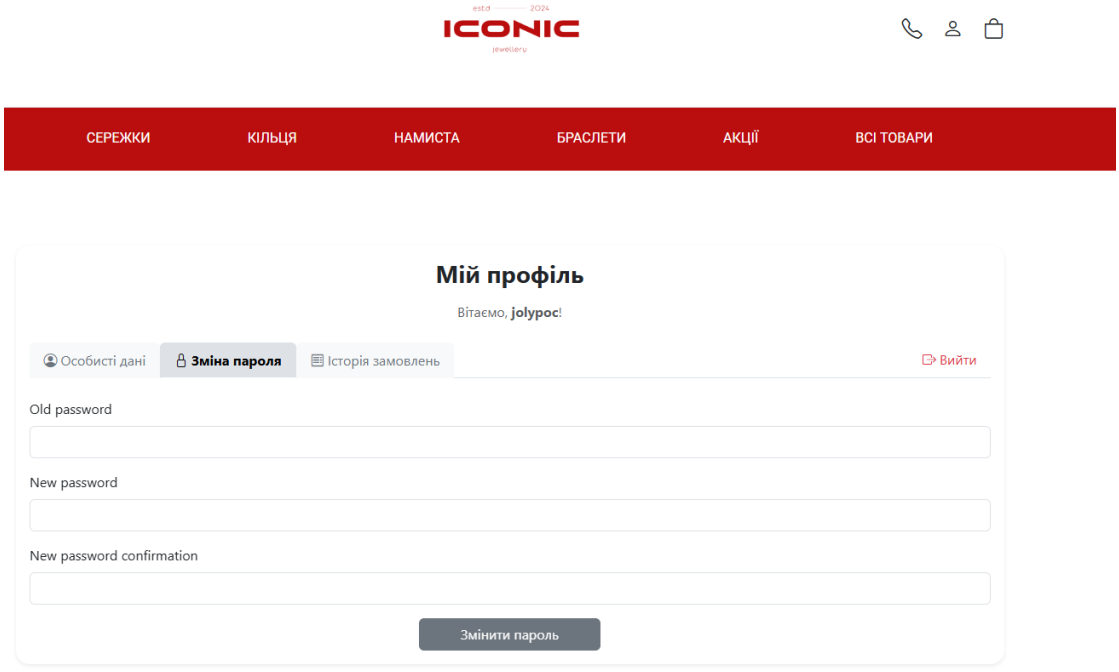


Рисунок 3.5 – Зміна паролю через особистий кабінет користувача

Також особистий кабінет містить розділ історії замовлень: в ньому виводиться список усіх замовлень, зроблених даним користувачем. Для отримання переліку замовлень використано запит до моделі замовлень з фільтрацією за поточним користувачем: Order.objects.filter(user=request.user). Таким чином, користувач бачить свої попередні замовлення (номер замовлення, дату, суму, статус, вміст замовлення) (див. рис. 3.6).

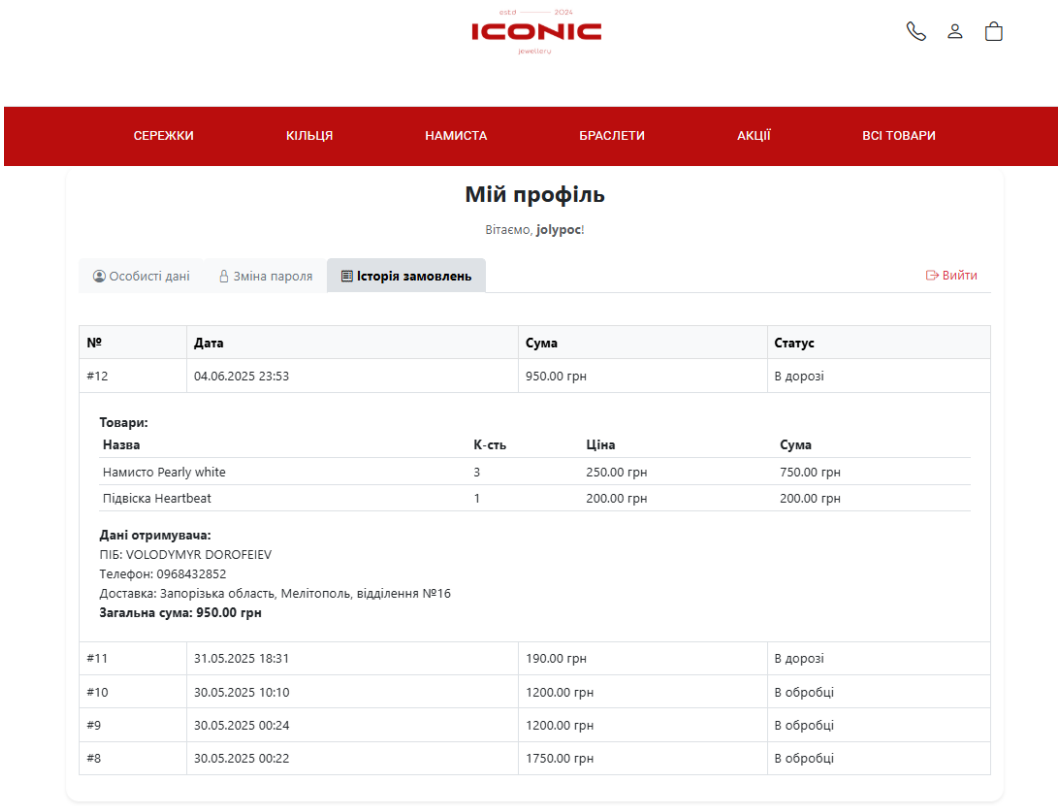


Рисунок 3.6 – Історія замовлень в особистому кабінеті

Для реалізації описаного функціоналу в системі використано представлення profile_view, яке обробляє запити до сторінки профілю користувача:

```
def profile_view(request):
    if request.method == 'POST':
        form_type = request.POST.get('form_type')
```

```
        if form_type == 'user_update':
            user_form = UserUpdateForm(request.POST, instance=request.user)
            password_form = PasswordChangeForm(request.user)
            if user_form.is_valid():
                user_form.save()
                messages.success(request, "Інформацію успішно оновлено.")
                return redirect('profile')
        elif form_type == 'password_change':
            user_form = UserUpdateForm(instance=request.user)
            password_form = PasswordChangeForm(request.user, request.POST)
            if password_form.is_valid():
                password_form.save()
                update_session_auth_hash(request, password_form.user)
                messages.success(request, "Пароль успішно змінено.")
                return redirect('profile')
    else:
        user_form = UserUpdateForm(instance=request.user)
        password_form = PasswordChangeForm(request.user)
    user_orders = Order.objects.filter(user=request.user).order_by('-created_at')
    return render(request, 'accounts/profile.html', {
        'user_form': user_form,
        'password_form': password_form,
        'user_orders': user_orders
    })
```

3.3.3 Захист доступу до персональних сторінок

Персональні сторінки користувача (профіль, історія замовлень, зміна пароля) мають бути захищені від несанкціонованого доступу. В реалізації цього проєкту для обмеження доступу використано декоратор Django login_required. Декоратор розміщується над функцією-представленням і автоматично перевіряє

статус автентифікації користувача. Якщо неавторизований користувач спробує звернутись до захищеного представлення, то login_required перенаправить його на сторінку входу (/accounts/login/), додавши параметр next з адресою, яку намагається відкрити користувач [31]. Після успішного входу за цим сценарієм користувач автоматично повертається на захищену сторінку. Якщо ж користувач вже увійшов у системо, декоратор дозволяє виконати представлення нормально. Таким чином, застосувавши @login_required можна гарантувати, що особисті дані та доступні для певного користувача функції (історія повідомлень, зміна паролю, редагування особистої інформації) будуть доступні тільки авторизованим особам.

Для забезпечення безпеки при зміні пароля використано стандартну форму Django – PasswordChangeForm. Ця форма вимагає введення поточного (старого) пароля користувача, нового пароля та підтвердження пароля. Вимога підтвердження старого пароля гарантує, що змінити пароль може лише той, хто вже знає діючий пароль (тобто власник облікового запису). Процес зміни пароля реалізований шляхом підключення стандартного класу PasswordChangeForm з django.contrib.auth. У випадку успішної зміни пароля користувача система оновлює облікові дані; подальший вхід до особистого кабінету можливий вже з використанням нового пароля.

Застосовані заходи (авторизаційни декоратори та верифікація старого пароля при зміні) відповідають сформованим вимогам до інформаційної системи. В результаті реалізовано надійний захист персональних сторінок: жодна конфіденційна інформація користувача не буде показана або змінена без попереднього входу в систему під його обліковим записом.

3.4 Реалізація каталогу товарів

3.4.1 Структура моделі товару

Модуль даних системи включає моделі Product, Category та ProductImage, які відповідають товарам, категоріям та зображенням товарів відповідно. Модель Category використовується для групування товарів за тематикою або типом, і містить такі поля як назва категорія та ідентифікатор. Зв’язок між категоріями і товарами реалізовано як «один до багатьох», так як до кожної категорії може входити велика кількість товарів, але один товар може належати лише до однієї категорії (сережки, кільця і.т.д). Для цього у моделі товару передбачено поле зовнішнього ключа category, що зберігає посилання на відповідну категорію.

Модель Product визначає основні атрибути товару: назву, опис, ціну та характеристики. Зокрема, передбачено поле price для базової ціни (типово DecimalField для точності), а також булеве поле is_discounted, яке позначає, чи застосована до товару знижка. Якщо поле is_discounted=True, то товар вважається акційним.

Модель ProductImage призначена для зберігання зображень, пов'язаних із конкретним товаром. Використання окремої моделі для фотографій товару забезпечує підтримку множинних зображень для одного товару (один товар – багато зображень). Модель містить поле зовнішнього ключа product, та поля для зберігання шляху до файлу зображення. Нижче наведено реалізацію моделей товару, категорії та зображення товару:

```
from django.db import models
class Category(models.Model):
    name = models.CharField(max_length=100)
    image = models.ImageField(upload_to='category_images/', blank=True, null=True)
    def __str__(self):
```

```
            return self.name
        class Product(models.Model):
            category = models.ForeignKey(Category, on_delete=models.CASCADE)
            name = models.CharField(max_length=200)
            description = models.TextField()
            characteristics = models.TextField(blank=True, null=True)
            price = models.DecimalField(max_digits=8, decimal_places=2)
            discount_price = models.DecimalField(max_digits=8, decimal_places=2,
blank=True, null=True)
            is_discounted = models.BooleanField(default=False)
            def get_price(self):
                return self.discount_price if self.is_discounted and self.discount_price else
self.price
            def __str__(self):
                return self.name
        class ProductImage(models.Model):
            product = models.ForeignKey(Product, on_delete=models.CASCADE,
related_name='images')
            image = models.ImageField(upload_to='product_images/')
            def __str__(self):
                return f"Image for {self.product.name}"
```

У зазначеному прикладі модель Product включає поле is_discounted для позначення акційних товарів та метод get_price(), що повертає або поле discount_price, або звичайну ціну. Модель Category несе інформацію про категорії товарів, а модель ProductImage зберігає записи про зображення товарів, пов’язані через зовнішній ключ із моделлю Product. Таким чином всі товари об’єднуються по відповідним категоріям товарі, а також реалізовано можливість додавати декілька фото на 1 товар.

3.4.2 Категорії товарів та фільтрація

Для зручності користувачів інтернет-магазину реалізовано можливість перегляду товарів за категоріями та застосування фільтрів. Зокрема, передбачено маршрут products_by_category, який відповідає за відображення сторінки каталогу товарів певної категорії. Цей маршрут отримує ідентифікатор категорії через сегмент URL та повертає список товарів, що належать до вибраної категорії. У контролері відбувається вибірка з бази даних за допомогою фільтрації по полю category_id. Таку вибірку у Django для моделі Product можна здійснити за допомогою методу filter():

```
from django.shortcuts import render, get_object_or_404
def products_by_category(request, category_id):
    category = get_object_or_404(Category, id=category_id)
    products = Product.objects.filter(category=category)
    categories = Category.objects.all()
    return render(request, 'products_by_category.html', {
        'current_category': category,
        'products': products,
        'categories': categories
    })
```

У наведеному коді функція products_by_category отримує із запиту category_id, завантажує відповідну категорію та вибирає всі товари, що їй належать. Одночасно отримується список усіх категорій – це дозволяє відобразити на сторінці навігаційне меню, що складається із списку категорій, по яким можна перейти. Сторінка products_by_category.html потім генерується з переданими даними: назвою поточної категорії та переліком товарів у ній.

Окрім розподілу товарів за категоріями, каталог підтримує додаткові режими відображення. Зокрема, реалізовано перегляд усіх товарів та окремо

акційних товарів. Сторінка «всі товари» (маршрут all_products) виводить повний список товарів незалежно від категорії: для цього в запиті не вказується жодний фільтр, і обираються всі записи моделі Product. Натомість сторінка «Акції» (маршрут discounted_products) відображає лише ті позиції, для яких is_discounted=True. Такий поділ дозволяє покупцю швидко переглянути всі акційні пропозиції окремо від загального списку.

Для реалізації каталогу товарів в системі використано окремі функції-обробники (views) для різних типів відображення – повний перелік товарів (all_products), акційні пропозиції (discounted_products) та товари певної категорії (products_by_category):

```
def product_detail(request, product_id):
    product = get_object_or_404(Product, pk=product_id)
    return render(request, 'products/product_detail.html', {'product': product})
def all_products(request):
    products = Product.objects.all()
    return render(request, 'products/all_products.html', {'products': products})
def discounted_products(request):
    products = Product.objects.filter(is_discounted=True)
    return render(request, 'products/discounts.html', {'products': products})
def products_by_category(request, category_id):
    category = get_object_or_404(Category, pk=category_id)
    products = Product.objects.filter(category=category)
    return render(request, 'products/products_by_category.html', {
        'category': category,
        'products': products
    })
def some_view(request):
    categories = Category.objects.all()
    return render(request, 'your_template.html', {'categories': categories})
```

```
def home(request):
    return render(request, 'home.html')
```

Кожна функція витягує відповідні об’єкти з бази даних за допомогою ORM-запитів і передає їх у свій окремий шаблон — all_products.html, discounts.html або products_by_category.html. Такий підхід дозволяє кастомізувати вигляд кожної сторінки відповідно до її призначення. Хоча альтернативою могло б бути використання одного універсального шаблону з параметризацією, окремі шаблони забезпечують вищу гнучкість у верстці. Таким чином, реалізована система каталогу підтримує як перегляд повного асортименту товарів, так і зручну фільтрацію за категоріями або акційними пропозиціями (див. рис. 3.7).

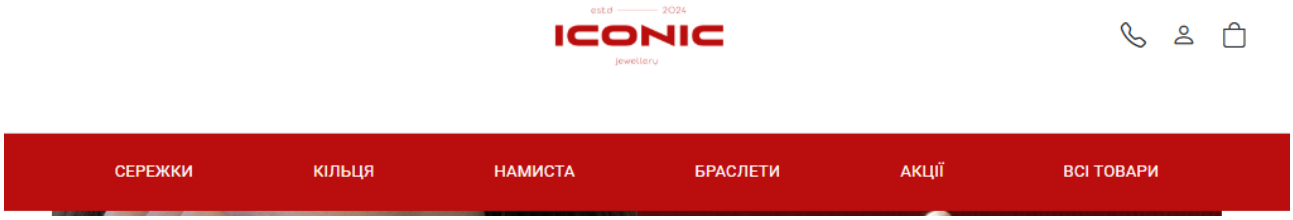


Рисунок 3.7 – Сортування та фільтрація товарів

3.4.3 Сторінка товару

Для відображення детальної інформації про конкретний товар реалізовано окрему функцію product_detail. Ця функція приймає ідентифікатор товару product_id як параметр і формує сторінку з повним описом товару. На відмінну від сторінки списку товарі, де показано багато позицій стисло, сторінка товару містить розгорнуті дані про один вибраний товар. Функція product_detail виконує вибірку моделі Product за заданим product_id та передає отриманий об’єкт у відповідний шаблон product_detail.html. Якщо товар з таким ідентифікатором не знайдений, повертається помилка 404 (товар відсутній). В шаблон також можуть

передаватися пов’язані об’єкти – наприклад, список зображень товару з моделі ProductImage. Нижче наведено код функції product_detail:

```
def product_detail(request, product_id):
    product = get_object_or_404(Product, pk=product_id)
    return render(request, 'products/product_detail.html', {'product': product})
```

У цьому коді функція get_object_or_404 з Django забезпечує отримання існуючого товару або генерування виключення (HTTP 404), якщо товар із заданим ідентифікатором відсутній у базі. Змінна product містить об’єкт товару з усіма його полями, доступними для використання у шаблоні. Додатково вибираються всі зображення, пов’язані з цим товаром, за допомогою фільтрації по зовнішньому ключу product. Зібрані дані передаються у шаблон product_detail.html для відображення. Шаблон сторінки товару (product_detail.html) будується таким чином, щоб представити користувачу всю релевантну інформацію про товар. Зокрема, в ньому відображається назва товару, його категорія, повний опис, актуальна ціна, а також галерея зображень товару. Ціна товару виводиться із використанням методу get_price(), що автоматично враховує наявність знижки – якщо товар акційний, показується нова знижена ціна поруч із старою ціною (якщо така зберігається). Галерея зображень генерується шляхом перебору списку images, переданого у контексті: кожне зображення відображається у вигляді елементу (наприклад, <img> з відповідним шляхом). Окрім цього, на сторінці товару присутні елементи для додавання товару в кошик. Сторінка детальної інформації про товар (див. рис. 3.8) є завершальною ланкою в структурі каталогу (див. рис 3.9): користувач, вибравши категорію, переглянувши список товарів і натиснувши на конкретний товар, потрапляє на його детальний опис.

Такий поетапний підхід відповідає класичному сценарію взаємодії з інтернет-магазином [32], де каталог товарів, сторінки категорій та сторінки товарів забезпечують інтуїтивну навігацію та повноту інформації для прийняття рішення про покупку.

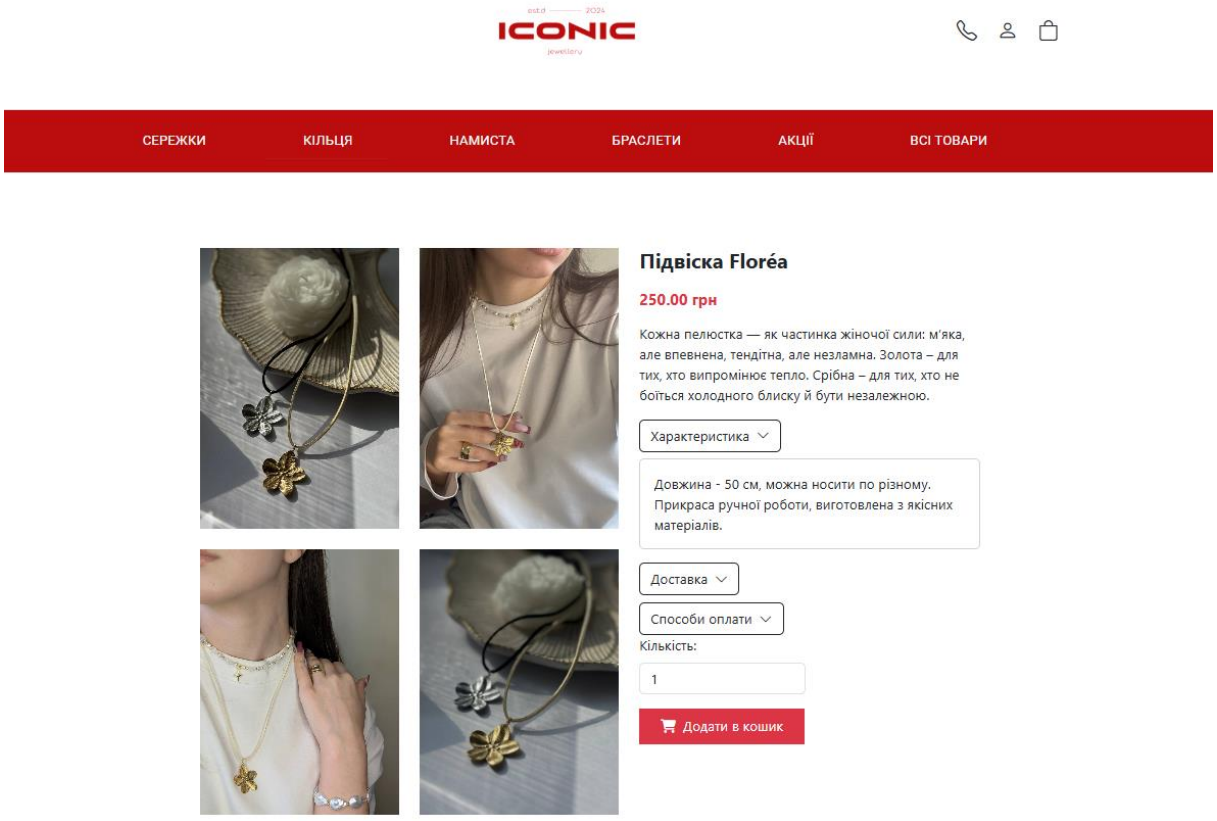


Рисунок 3.8 – Сторінка товару

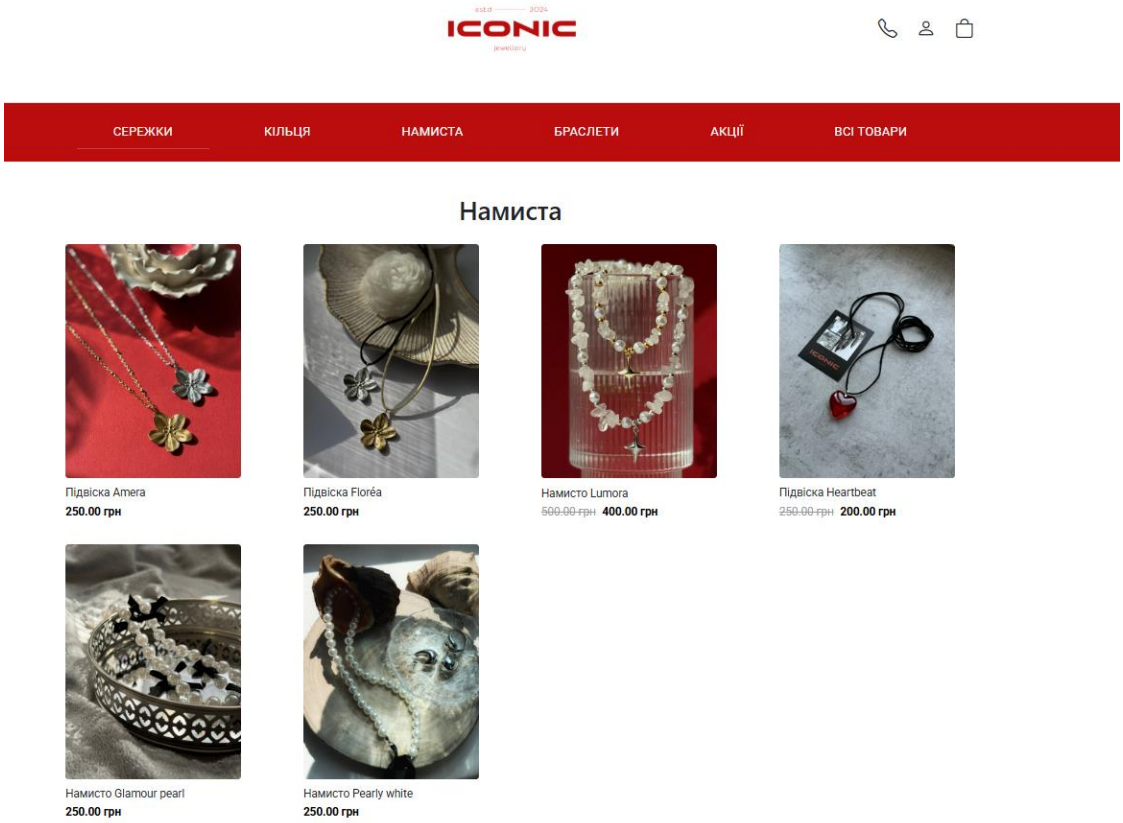


Рисунок 3.9 – Каталог товару

3.5 Реалізація кошика та оформлення замовлення

Для того, щоб клієнт зміг придбати декілька товарів за один раз, в інформаційній системі реалізовано функцію кошика для того, щоб клієнт зміг обрати певну кількість товарів і в подальшому редагувати своє майбутнє замовлення [33]. У розробленому веб-магазині на Django кошик реалізовано як набір елементів CartItem, прив'язаних до сесії користувача. Це означає, що дані кошика зберігаються на стороні сервера для кожного відвідувача окремо (ідентифікуються за унікальним ID сесії) [34]. Такий підхід дозволяє додавати товари до кошика і не вимагати авторизації до моменту оформлення замовлення. Далі розглянуто основні етапи роботи кошика: додавання товарів, перегляд/редагування вмісту, безпосередньо оформлення замовлення та відображення підтвердження успіху замовлення.

Додавання товару до кошика відбувається при натисканні користувачем кнопки «Додати в кошик» біля обраного товару. Ця дія викликає відповідну функцію у views, яка обробляє запит типу POST. Наприклад, функція add_to_cart(request, product_id) отримує ідентифікатор продукту, знаходить його в базі даних і додає до сесійного кошика. Якщо цей товар уже є в кошику, його кількість збільшується, інакше товар додається як новий елемент [35]. Після оновлення даних користувача перенаправляють на сторінку кошика, де він може побачити актуальний список вибраних товарів (див. рис. 3.10).

Для забезпечення коректності операції використано декоратор @require_POST, щоб додавати товари до кошика виключно через метод POST:

```
from django.views.decorators.http import require_POST
from django.shortcuts import redirect, get_object_or_404, render
from .models import CartItem
from products.models import Product
from django.contrib import messages
def add_to_cart(request, product_id):
```

```
    product = get_object_or_404(Product, id=product_id)
    quantity = int(request.POST.get('quantity', 1))
    session_key = request.session.session_key
    if not session_key:
        request.session.create()
        session_key = request.session.session_key
    item, created = CartItem.objects.get_or_create(
        product=product,
        session_key=session_key,
        defaults={'quantity': quantity}
    )
    if not created:
        item.quantity += quantity
        item.save()
```

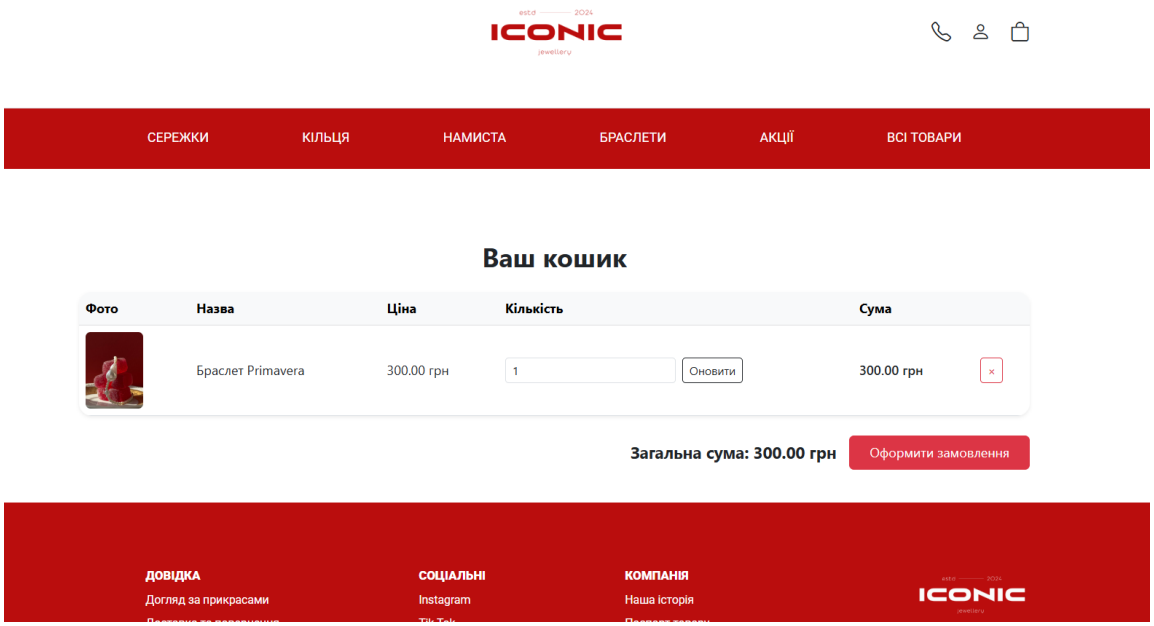


Рисунок 3.10 – Сторінка кошика

У наведеному прикладі додавання товару до кошика здійснюється через функцію add_to_cart, яка обробляє POST-запит із ідентифікатором товару. У цій реалізації використовується модель CartItem, яка зберігає товари безпосередньо

в базі даних. Прив’язка до користувача здійснюється через ключ сесії (якщо користувач неавторизований) або через ForeignKey до облікового запису (якщо авторизований).

Функція працює за наступною логікою:

– спочатку за переданим product_id знаходиться відповідний товар у базі;
– далі визначається поточна сесія користувача. Якщо сесія ще не існує (тобто користувач зайшов уперше), вона створюється автоматично за допомогою request.session.create();
– потім виконується пошук товару в таблиці CartItem за поточним ключем сесії: якщо запис уже існує – просто оновлюється його кількість; якщо ні — створюється новий запис із заданою кількістю;
– такий підхід дозволяє зберігати вміст кошика навіть для неавторизованих користувачів – допоки активна сесія.

Таким чином, реалізація забезпечує гнучку логіку додавання товарів до кошика, не потребуючи окремого класу для збереження в request.session. Це також спрощує подальший перегляд вмісту кошика, оскільки всі позиції вже зберігаються у відповідній моделі бази даних.

Коли всі потрібні товари додано, а вміст кошика відредаговано, користувач переходить до оформлення замовлення (див. рис. 3.11).

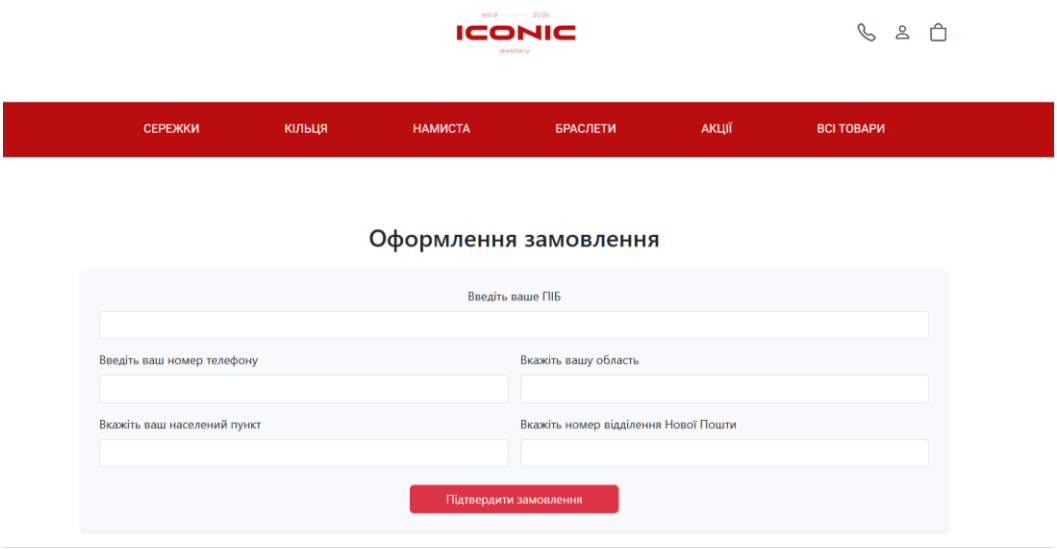


Рисунок 3.11 – Оформлення замовлення

Користувач заповнює поля форми, зокрема ПІБ, телефон та адресу доставки. Після заповнення форми й натискання кнопки підтвердження відбувається обробка замовлення на сервері.

```
def place_order(request):
    session_key = request.session.session_key
    if not session_key:
        request.session.create()
        session_key = request.session.session_key
    cart_items = CartItem.objects.filter(session_key=session_key)
    if not cart_items.exists():
        return redirect('cart')
    if request.method == 'POST':
        form = OrderForm(request.POST)
        if form.is_valid():
            if request.user.is_authenticated:
                order.user = request.user
            order.save()
            for item in cart_items:
                OrderItem.objects.create(
                    order=order,
                    product=item.product,
                    quantity=item.quantity
                )
            cart_items.delete()
            return redirect('order_success')
    else:
        form = OrderForm()
    return render(request, 'orders/place_order.html', {
        'form': form,
```

```
        'items': cart_items
    })
def order_success(request):
    return render(request, 'orders/success.html')
```

Обробка форми замовлення здійснюється через форму OrderForm. Якщо дані, введені користувачем, проходять валідацію, створюється об’єкт замовлення Order. При цьому об’єкт ще не зберігається в базі одразу (параметр commit=False), що дозволяє попередньо додати зв’язок з обліковим записом користувача, якщо він авторизований. Після збереження замовлення до нього додаються всі товари, які користувач вибрав у кошику. Для кожного товару створюється об’єкт OrderItem з інформацією про товар і кількість. Після обробки і збереження даних кошик очищується методом cart_items.delete(), щоб уникнути дублювання під час повторного відкриття сторінки. Після успішного збереження замовлення користувач отримує візуальне підтвердження, що покупку завершено. У нашому випадку користувача перенаправляють на сторінку підтвердження (див. рис. 3.12).

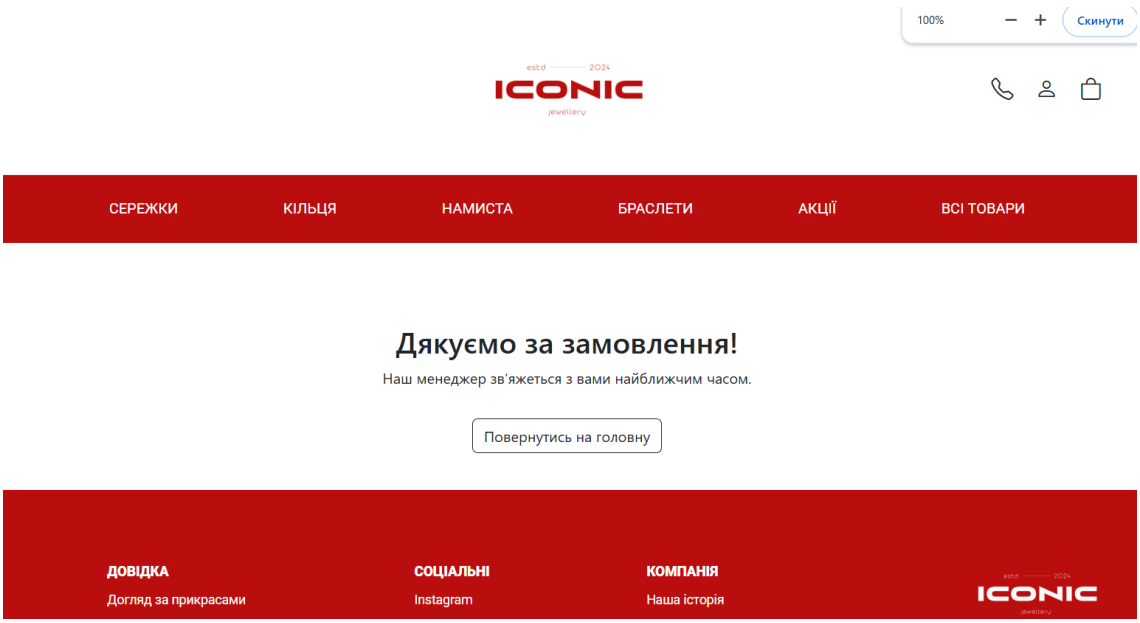


Рисунок 3.12 – Повідомлення про успішне замовлення

Це повідомлення формується на стороні сервера після створення замовлення – як було показано у коді вище, використано функцію order_success для додавання флеш-повідомлення про успішне замовлення. Таким чином, система інформує клієнта, що замовлення прийнято до обробки, і завершує процес оформлення замовлення.

3.6 Висновки до розділу 3

У третьому розділі було детально розглянуто с програмної реалізації інформаційної системи для онлайн-продажу біжутерії. Було створено повнофункціональний вебсайт з особистим кабінетом користувача, каталогом товарів, кошиком та механізмом оформлення замовлення.

Зокрема, реалізовано:

- реєстрацію та авторизацію користувачів, що забезпечує захищений доступ до персональних сторінок та можливість перегляду історії замовлень;
- особистий кабінет, де користувач може змінювати персональні дані, редагувати пароль і переглядати зроблені замовлення;
- каталог товарів із фільтрацією, який дозволяє групувати продукцію за категоріями та відображати акційні позиції та каталог всіх товарів;
- сторінку товару з інтерфейсом замовлення, що включає візуальне оформлення, опис, характеристики, фото та можливість додати товар у кошик;
- функціонал кошика, який зберігає вибрані товари у сесії навіть для неавторизованих користувачів;
- механізм оформлення замовлення, що передбачає збереження даних клієнта, формування списку товарів, очищення кошика та повідомлення про успішне оформлення;

Усі програмні компоненти реалізовано з використанням фреймворку Django, який забезпечив ефективну роботу з базою даних, маршрутизацію, обробку форм і шаблонів. Результатом стала система, яка відповідає сучасним вимогам до e-commerce ресурсів, поєднуючи функціональність і зручність використання як для користувача, так і для адміністратора.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було розроблено інформаційну систему для онлайн-реалізації біжутерії з особистим кабінетом користувача. Робота охопила всі основні етапи створення вебсистеми — від аналізу предметної області до безпосередньої реалізації функціоналу та перевірки працездатності розробленого продукту.

На основі аналізу ринку онлайн-торгівлі було визначено ключові вимоги до інформаційної системи, сформульовано функціональні та нефункціональні характеристики, а також побудовано загальну архітектуру рішення. Для проєктування системи застосовано сучасні методи моделювання — створено діаграми варіантів використання, структури даних, взаємодії між компонентами.

Програмну реалізацію виконано з використанням фреймворку Django (Python) та СУБД SQLite, що дозволило забезпечити ефективну інтеграцію моделей, обробку запитів і зручне управління даними. В системі реалізовано такі ключові компоненти:

- функціонал реєстрації та авторизації користувачів;
- особистий кабінет із переглядом історії замовлень та зміною персональних даних;
- каталог товарів з категоріями, фільтрацією та сторінками деталей;
- кошик, що зберігає дані через сесію;
- механізм оформлення замовлень із збереженням даних у базі.

Сайт має адаптивний інтерфейс, розроблений із використанням HTML, CSS (Bootstrap) та динамічних елементів, що забезпечують зручність для кінцевого користувача.

Розроблена інформаційна система відповідає поставленим вимогам, є масштабованою та може бути використана як основа для комерційного запуску онлайн-магазину. Отримані результати свідчать про практичну значущість розробки й можливість її подальшого розширення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Інформаційна система. Вікіпедія – вільна енциклопедія. URL : https://w.wiki/8Nni (дата звернення : 13.04.2025).

2. Створення сайту з особистим кабінетом. WebCase. URL : https://webcase.com.ua/uk/razrabotka-lichnogo-kabineta/ (дата звернення : 14.04.2025).

3. Vans. Вікіпедія – вільна енциклопедія. URL : https://w.wiki/E2$j (дата звернення : 16.04.2025).

4. Списки бажань: чому без них не обійтися інтернет-магазину. eSputnik. URL : https://esputnik.com/uk/blog/spiski-bazhan-chomu-bez-nih-ne-obijtisya-internet-magazinu (дата звернення : 16.04.2025).

5. Що можна продавати в інтернеті: 20+ ідей для бізнесу. SendPulse. URL : https://sendpulse.ua/blog/what-to-sell-online (дата звернення : 17.04.2025).

6. Як відкрити інтернет-магазин косметики та парфумерії? Brainlab. URL : https://brainlab.com.ua/uk/blog-uk/yak-vidkriti-internet-magazin-kosmetiki-ta-parfumeriyi (дата звернення : 18.04.2025).

7. Дешевий і агресивний. Мережа одягу Sinsay. Forbes. URL : https://forbes.ua/company/desheviy-i-agresivniy-merezha-odyagu-sinsay-vidkrila-mayzhe-50-magaziniv-u-2024-mu-v-planakh-do-kintsya-roku-shche-100-chim-tse-nebezpechno-dlya-ukrainskogo-rinku-14082024-22999 (дата звернення : 25.04.2025).

8. Як відкрити магазин біжутерії з нуля: Інструкція для українських підприємців. Business broker. URL : https://business-broker.com.ua/blog/iak-vidkryty-mahazyn-bizhuterii-z-nulia-instruktsiia-dlia-ukrainskykh-pidpryiemtsiv/ (дата звернення : 25.04.2025).

9. Як створити інтернет магазин для продажу прикрас. Brainlab. URL : https://brainlab.com.ua/uk/blog-uk/yak-stvoriti-internet-magazin-dlya-prodazhu-prikras (дата звернення : 29.04.2025).

10. Денис Демчина. Як відкрити магазин біжутерії з нуля: Інструкція для українських підприємців. Business-broker. URL : https://business-broker.com.ua/blog/iak-vidkryty-mahazyn-bizhuterii-z-nulia-instruktsiia-dlia-ukrainskykh-pidpryiemtsiv (дата звернення : 29.04.2025).

11. Software Requirements. Tutorials Point. Tutorialspoint. URL : https://www.tutorialspoint.com/software_engineering/software_requirements.htm (дата звернення : 02.05.2025).

12. Classification of Software Requirements – Software Engineering. Geeks for Geeks. URL : https://www.geeksforgeeks.org/software-engineering-classification-of-software-requirements/ (дата звернення : 03.05.2025).

13. Functional Requirements. Geeks for Geeks. URL : https://www.geeksforgeeks.org/software-engineering-classification-of-software-requirements/ (дата звернення : 05.05.2025).

14. Підказки щодо написання технічного завдання на розробку інтернет-магазину. Seo Evolution. URL : https://seo-evolution.com.ua/blog/razrabotka/napisannya-tehnichnogo-zavdannya-na-rozrobku-internet-magazinu (дата звернення : 06.05.2025).

15. Формалізація. Вікіпедія – вільна енциклопедія. URL : https://w.wiki/EEv6 (дата звернення : 09.05.2025).

16. Класифікація інформаційних систем. Studfile. URL : https://studfile.net/preview/5118185/page:12/ (дата звернення : 10.05.2025).

17. Архітектура програмного забезпечення. Artofba. URL : https://www.artofba.com/uk/post/main-types-of-software-architecture (дата звернення : 15.05.2025).

18. Переваги та недоліки різних підходів до архітектури. Wezom. URL : https://wezom.com.ua/ua/blog/arhitektura-programnogo-zabezpechennya (дата звернення : 16.05.2025).

19. Модель-вид-контролер. Вікіпедія – вільна енциклопедія. URL : https://w.wiki/DmKE (дата звернення : 16.06.2025).

20. Найважливіші архітектурні шаблони, які необхідно знати. Devzone. URL : https://devzone.org.ua/post/nayvazlyvishi-arkhitekturni-shablony-iaki-neobkhidno-znaty#:~:text= (дата звернення : 16.05.2025).

21. Інформаційна модель. Вікіпедія – вільна енциклопедія. URL : https://w.wiki/EMGK (дата звернення : 17.05.2025).

22. Поліна S. Лекція – Інформація і система. Інформаційні моделі систем. Scribd. URL : https://www.scribd.com/document/684101715 (дата звернення : 18.05.2025).

23. Правильна структура інтернет-магазину: основні етапи розробки. Elit-web. URL : https://elit-web.ua/ua/blog/pravilnaja-struktura-internet-magazina (дата звернення : 19.05.2025).

24. Top 10 Python Web Development Frameworks in 2025. Browserstack. URL : https://www.browserstack.com/guide/top-python-web-development-frameworks (дата звернення : 19.05.2025).

25. Django Vs. Flask: Understanding The Major Differences. Simplilearn. URL : https://www.simplilearn.com/flask-vs-django-article (дата звернення : 20.05.2025).

26. Introduction to SQLite. Geeks for Geeks. URL : https://www.geeksforgeeks.org/introduction-to-sqlite/ (дата звернення : 20.05.2025).

27. PHP vs. Python: How to Choose the Right Programming Language. Mobilunity. URL : https://mobilunity.com/blog/php-vs-python/ (дата звернення : 20.05.2025).

28. Django vs Laravel: Which Is the Better Framework? Kinsta. URL : https://kinsta.com/blog/django-vs-laravel/ (дата звернення : 20.05.2025).

29. Ravindran, A. Django Design Patterns and Best Practices. Birmingham: Packt Publishing, 2015. 222 p.

30. Django Software Foundation. Writing your first Django app, part 1. Djangoproject. URL : https://docs.djangoproject.com/en/5.2/intro/tutorial01/ (дата звернення : 21.05.2025).

31. Jim McGaw. Beginning Django E-Commerce. Archive. URL : https://archive.org/stream/BeginningDjangoECommerce/Beginning+Django+E-Commerce_djvu.txt (дата звернення : 21.05.2025).

32. Holovaty A., Jacob K. Moss. The Definitive Guide to Django: Web Development Done Right. 358 p.

33. Григор’єва М. І. Міжнародна електронна комерція. Київ, 2016. 65 с.

34. Django Software Foundation. How to use sessions. Django documentation. URL : https://docs.djangoproject.com/en/5.2/topics/http/sessions/ (дата звернення : 24.05.2025).

35. Кудь В. С. Створення онлайн магазину на Python Django : кваліфікаційна робота бакалавра / В. С. Кудь ; наук. керівник О. В. Сидоренко. Державний університет телекомунікацій, 2024. 94 с. URL : https://duikt.edu.ua/repozitorii/ki/2024/бак/Диплом_Кудь.pdf (дата звернення : 26.05.2025).