ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

КВАЛІФІКАЦІЙНА РОБОТА

Тема: «Вебсайт інтернет-магазину традиційного одягу із використанням алгоритмів рекомендацій та оптимізації цін»

Ступінь вищої освіти – бакалавр
Спеціальність – 122 «Комп'ютерні науки»
Освітня програма «Комп'ютерні науки»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: здобувач 4 курсу
групи КН-22
Ростислав ОРЛОВСЬКИЙ
Керівник: викладач кафедри інформаційного менеджменту, математики та статистики
Олег МУШИНСЬКИЙ

Засвідчую, що кваліфікаційна робота оформлена відповідно до ДСТУ 3008:2015 та не містить запозичень з праць інших авторів без відповідних посилань.
Здобувач: _____
(підпис)

м. Київ – 2026 рік

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

"УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА "КРОК"

ЗАТВЕРДЖУЮ:

викладач кафедри інформаційного

менеджменту, математики та статистики

_____ Сергій МІЧКІВСЬКИЙ

«___» _____20___ р

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Орловський Ростислав Андрійович

Тема роботи	Вебсайт інтернет-магазину традиційного одягу із використанням алгоритмів рекомендацій та оптимізації цін
Номер та дата наказу про затвердження теми	№ 133-7 від 22 грудня 2025 року
Коротка постановка завдання	Розробка вебсайту для інтернет-магазину традиційного одягу із використанням алгоритмів рекомендацій та оптимізації цін
Посилання на джерела інформації (не більше п’яти найменувань, які рекомендує науковий керівник)	1. Ліман В., Малініч І., Малініч П. Перспективи використання штучного інтелекту для ціноутворення в інтернет-торгівлі. Вимірювальна та обчислювальна техніка в технологічних процесах. 2024. Вип. 2. С. 325–330. DOI: https://doi. org/10.31891/2219-9365-2024-78-37 2. Django Web Framework (Python) / MDN Web Docs. Mozilla Foundation, 2026. URL: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django (дата звернення: 08.04.2026).
Вимоги до кваліфікаційної роботи	Кваліфікаційна робота має передбачити теоретичне, системотехнічне або експериментальне дослідження складного спеціалізованого завдання або практичної проблеми в галузі комп’ютерних наук, яке характеризується комплексністю та невизначеністю умов і потребує застосування теорій і методів інформаційних технологій.

Дата видачі завдання 27 грудня 2025 р.

Керівник Олег МУШИНСЬКИЙ

Здобувач освітнього ступеня бакалавра Ростислав ОРЛОВСЬКИЙ

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання	Примітка
Підготовчий етап			
1	Вибір напряму дослідження	01.12.2025 р.	Виконано
2	Формування теми та призначення керівника	15.12.2025 р.	Виконано
3	Затвердження теми кваліфікаційної роботи	22.12.2025 р.	Виконано
4	Затвердження завдання на кваліфікаційну роботу	26.12.2025 р.	Виконано
Основний етап			
5	Розробка концепції кваліфікаційної роботи	23.01.2026 р.	Виконано
6	Підбір та вивчення джерел інформації з напряму дослідження. Огляд існуючих аналогів	13.03.2026 р.	Виконано
7	Затвердження розширеної постановки завдання. Підготовка та подання керівникові розділу 1 кваліфікаційної роботи	20.03.2026 р.	
8	Проєктування. Підготовка та подання керівникові розділу 2 кваліфікаційної роботи	27.03.2026 р.	
9	Підготовка доповіді для експертизи стану виконання кваліфікаційної роботи (проміжний контроль)	30.03-04.04.2026 р.	
10	Реалізація. Підготовка та подання керівникові розділу 3 кваліфікаційної роботи	06.04.2026 р.	
11	Підготовка та подання керівнику першого варіанту всієї кваліфікаційної роботи	13.04.2026 р.	
12	Доопрацювання кваліфікаційної роботи з урахуванням зауважень керівника та представлення керівникові доопрацьованого варіанту кваліфікаційної роботи	20.04.2026 р.	
Завершальний етап			
13	Представлення рукопису для перевірки на плагіат	27.04-03.05.2026 р.	
14	Підготовка презентації та доповіді на передзахист	04.05-10.05.2026 р.	
15	Передзахист кваліфікаційної роботи	11.05-15.05.2026 р.	
16	Доопрацювання роботи за результатами передзахисту	18.05-05.06.2026 р.	
17	Експертиза роботи керівником та зовнішнім експертом	08.06-14.06.2026 р.	
18	Доопрацювання доповіді та презентації для захисту	08.06-14.06.2026 р.	
19	Захист кваліфікаційної роботи	15.06-21.06.2026 р.	

Керівник Олег МУШИНСЬКИЙ

Здобувач освітнього ступеня бакалавра Ростислав ОРЛОВСЬКИЙ

АНОТАЦІЯ

Орловський Р.А. Вебсайт інтернет-магазину традиційного одягу із використанням алгоритмів рекомендацій та оптимізації цін

Пояснювальна записка кваліфікаційної роботи за спеціальністю 122 – Комп'ютерні науки (освітня програма – Комп'ютерні науки) СО Бакалавр. – ВНЗ «Університет економіки та права «КРОК», Навчально-науковий інститут інформаційних та комунікаційних технологій, кафедра комп'ютерних наук, Київ, 2026.

У даній роботі розроблено вебсайт інтернет-магазину з інтеграцією алгоритму колаборативної фільтрації (k-найближчих сусідів) для персоналізованих рекомендацій та моделі цінової еластичності попиту (PED) для динамічної оптимізації цін.

Ключові слова: інтернет-магазин, електронна комерція, колаборативна фільтрація, динамічне ціноутворення, Django, Celery.

Табл. 1. Малюнок. 16. Бібліограф. 22. Формули 2.

Orlovskyi R.A. Website of an online store for traditional clothing using recommendation algorithms and price optimization.

Explanatory note of qualification work in specialty 122 - Computer Science (educational programme - Computer Science), Bachelor's degree - University of Economics and Law "KROK", Educational and Research Institute of Information and Communication Technologies, Department of Computer Science, Kyiv, 2026.

In this work, a traditional clothing online store website has been developed, integrating a collaborative filtering algorithm (k-nearest neighbors) for personalized recommendations and a Price Elasticity of Demand (PED) model for dynamic price optimization.

Keywords: online store, e-commerce, collaborative filtering, dynamic pricing, Django, Celery. Table 1. Fig. 16. Bibliography. 22. Formula 2.

ЗМІСТ

ВСТУП

Актуальність теми. В Україні від 2022 спостерігається стрімкий сплеск інтересу до традиційного українського одягу. Для існуючих фізичних точок продажу перехід у цифровий простір є необхідною умовою для масштабування бізнесу. Висока конкуренція вимагає не просто створення сайту із каталогом товарів, а впровадження інтелектуальних систем, що здатні персоналізувати клієнтський досвід та оптимізувати бізнес-процеси через автоматизоване ціноутворення.

Мета дослідження. Розробка вебсайту інтернет-магазину традиційного одягу із використанням алгоритмів рекомендацій та оптимізації цін.

Завдання дослідження:

– проаналізувати предметну область інтернет-торгівлі та специфіку ринку традиційного одягу;

– дослідити існуючі методи та алгоритми рекомендаційних систем та динамічного ціноутворення;

– спроєктувати архітектуру вебсайту та структуру бази даних;

– розробити програмне забезпечення інтернет-магазину з інтегрованими модулями;

– провести тестування розроблених алгоритмів та оцінити їхню ефективність;

Об’єкт дослідження. Процеси онлайн-продажів, формування персоналізованих пропозицій та управління вартістю товарів в електронній комерції.

Предмет дослідження. Розробка вебсайту інтернет-магазину традиційного одягу, алгоритми рекомендацій на основі вподобань користувачів та методи оптимізації цін залежно від ринкових показників.

Методи дослідження. Для розв'язання поставлених у роботі завдань було використано комплексний підхід, що базується на поєднанні таких методів:

- теоретичні методи: системний аналіз, узагальнення та порівняльний аналіз науково-технічних джерел - для дослідження предметної області, вивчення аналогів системи та обґрунтування її архітектури;
- методи моделювання: об'єктно-орієнтований аналіз та проєктування (ООАП) - для побудови структури бази даних та логічної схеми взаємодії компонентів вебплатформи;
- спеціальні математичні методи: методи колаборативної фільтрації та аналізу еластичності попиту - для розробки інтелектуальних алгоритмів рекомендацій та стратегій динамічного ціноутворення;
- емпіричні методи: комп'ютерний експеримент та функціональне тестування - для практичної перевірки працездатності створеного програмного прототипу та верифікації результатів обчислень.

Практичне значення. Розроблений вебсайт інтернет-магазину дозволяє перейти фізичному бізнесу в онлайн-сегмент, забезпечуючи покращений підхід до роботи з клієнтами та автоматизацію бізнес-процесів.

Структура роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків та списку посилань (15 найменувань). Пояснювальна записка містить 15 рисунки. Загальний обсяг пояснювальної записки складає 50 сторінок, основний зміст викладено на сторінках 45.

РОЗДІЛ 1
ПОСТАНОВКА ЗАВДАННЯ НА РОЗРОБКУ ВЕБ-САЙТУ ІНТЕРНЕТ-МАГАЗИНУ ІЗ МОДУЛЯМИ РЕКОМЕНДАЦІЙ ТА ОПТИМІЗАЦІЇ ЦІН

1.1 Опис предметної області

Сучасний ринок традиційного українського одягу трансформувався з нішевого сувенірного сегмента в елемент повсякденного та преміального гардеробу. Кожна одиниця має складну систему атрибутів: тип полотна, техніка вишивки, регіональні особливості орнаменту, символіка, тощо; все це створює високе когнітивне навантаження на покупця при виборі.

Цільова аудиторія динамічно розширюється за рахунок внутрішнього попиту та значної кількості українців за кордоном, які використовують етно-одяг як засіб самоідентифікації. Об’єктом алгоритмічної оптимізації у роботі є інформаційні процеси взаємодії з клієнтами та формування вартості у спеціалізованому магазині етно-одягу, що поєднує роздрібні фізичні точки із системою онлайн-дистриб'юції.

Основні бізнес-процеси:

– прямі продажі із взаємодією продавець-клієнт, де рекомендації надаються суб’єктивно;

– облік товарів на складі;

– формування ціни, яке базується на собівартості та націнки, без реагування на зміну попиту в цифровому середовищі.

Предметна область охоплює сферу електронної комерції (e-commerce), зокрема процеси цифрової дистриб'юції товарів, алгоритмічного керування товарними каталогами та інтелектуального аналізу поведінки інтернет-споживачів.

Рекомендаційні системи у цьому випадку виконують роль цифрового консультанта, аналізуючи дані про вподобання клієнтів та пропонуючи

найбільш релевантні товари, що підвищує шанс їх придбання. Процес автоматичного коригування цін на основі алгоритмів, що враховують залишки та популярність товару, дозволяють нівелювати ризики затоварення складу та максимізувати маржинальність кожного замовлення.

1.2 Визначення потенційних конкурентів

Etnodim - український бренд сучасного етно-одягу, що поєднує традиційні орнаменти з актуальними формами [3]. Вебсайт компанії слугує головним майданчиком для презентації колекцій та взаємодії з клієнтами по всьому світу. Основною метою ресурсу є створення цифрового простору, де покупець може ознайомитися з філософією бренду та придбати продукцію. Користувачі можуть переглядати детальні каталоги, фільтрувати товари за категоріями та використовувати персональний кошик для оформлення замовлень. Проведено аналіз сайту і визначено переваги та недоліки.

Переваги:

– високий рівень візуалізації: використання професійного контенту та продуманого UX/UI дизайну забезпечує занурення користувача в естетику бренду;

– наявність блогу та детальних описів символіки орнаментів допомагає клієнту зробити усвідомлений вибір, що підвищує лояльність до продукту.

Недоліки:

– статичність рекомендацій: на сайті відсутні блоки з рекомендаціями, товар відображається у статичному каталозі. “Товари у тому ж стилі” має суворо визначений перелік (рис. 1.1);

– система не передбачає автоматичного коригування цін залежно від коливань попиту, що обмежує можливості максимізації прибутку в періоди пікової популярності товарів.

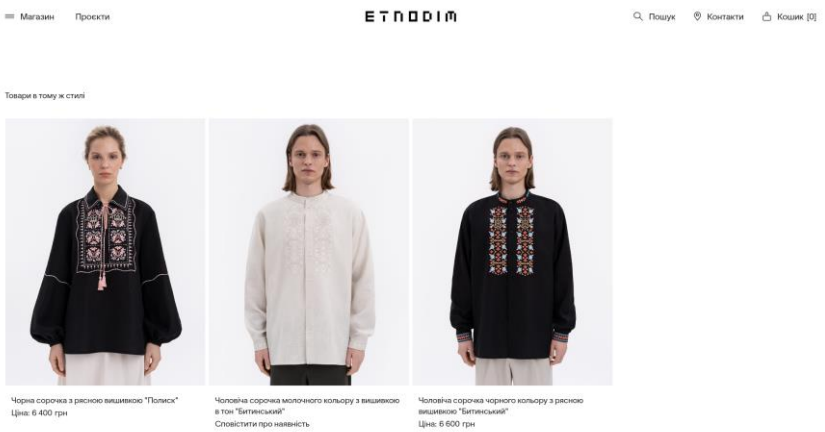


Рисунок 1.1 - Зображення блоку "Товари у тому ж стилі" на сторінці товару.

Джерело: Etnodim [3]

Vilni (рис. 1.2) - концептуальний магазин традиційного одягу, орієнтований на молодіжну аудиторію [4]. Платформа побудована на базі готового хмарного рішення (SaaS [5]), що дозволяє бренду швидко оновлювати асортимент. Проведено аналіз сайту і визначено переваги та недоліки.

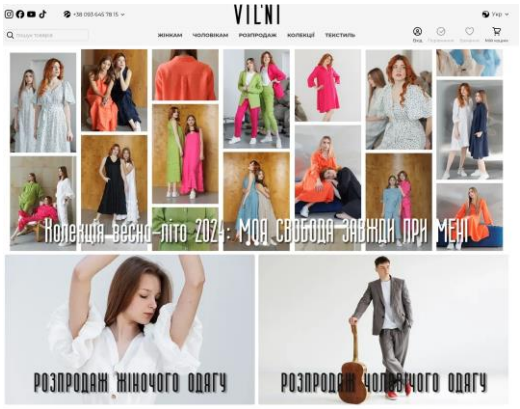


Рисунок 1.2 - Зображення головної сторінки Vilni.

Джерело: Vilni [4].

Переваги:

– сайт демонструє високу швидкість завантаження сторінок;

– інтерфейс коректно відображається на всіх типах мобільних пристроїв, що є критичним для сучасної електронної комерції.

Недоліки:

– обмеженість алгоритмів: використання стандартної платформи не дозволяє впроваджувати складні математичні моделі персоналізації, обмежуючи рекомендації лише базовою фільтрацією за категоріями;

– система не здатна аналізувати глибинні атрибути вишивки (техніку, регіон) для пропозиції релевантних доповнень до образу (пропонувати додатковий товар).

Rozetka - найбільший універсальний маркетплейс України, який займає значну частку ринку в сегменті одягу [6]. Платформа надає інструменти для тисяч продавців, забезпечуючи потужний функціонал пошуку та фільтрації за ціною та брендом. Проведено аналіз сайту і визначено переваги та недоліки.

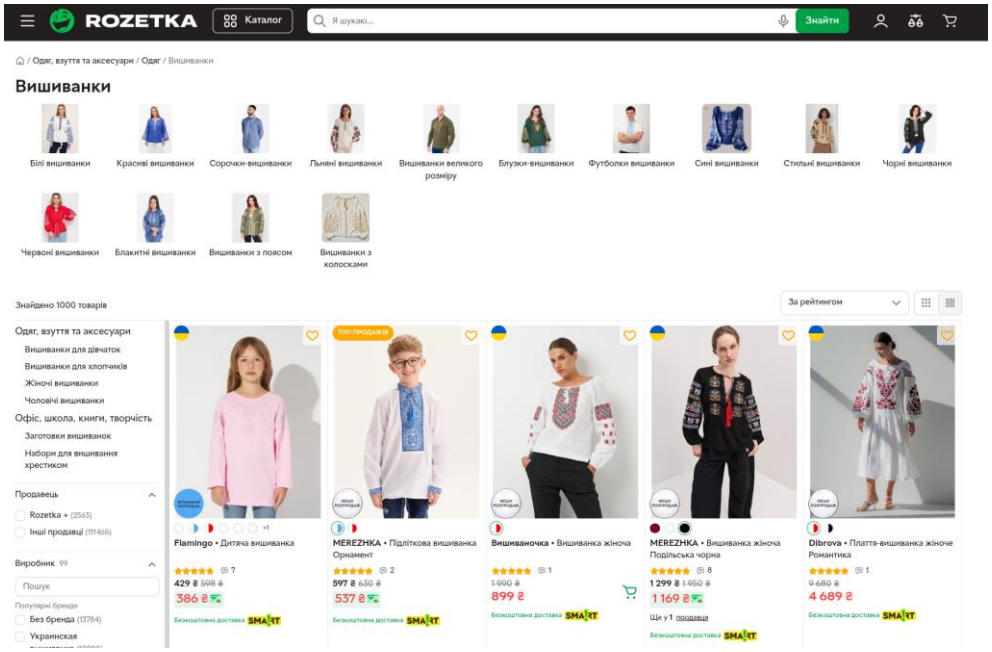


Рисунок 1.3 - Зображення категорії "Вишиванки" на маркетплейсі. Джерело: Rozetka [6].

Переваги:

– велика кількість користувачів дозволяє системі накопичувати статистику для базових рекомендацій типу «з цим товаром купували»;

– наявність відпрацьованих механізмів відгуків, рейтингів та систем лояльності, що підвищує довіру покупця до процесу транзакції.

Недоліки:

– алгоритми рекомендацій на платформі є універсальними і не враховують специфічні метадані етно-одягу, що призводить до низької релевантності рекомендацій у цій категорії;

– відсутність інструментів для продавця: малі бренди не мають доступу до автоматизованого динамічного ціноутворення, через що ціни залишаються статичними незалежно від локальних трендів попиту.

На основі виявлених недоліків комерційних аналогів було сформовано перелік концептуальних переваг та функціональних особливостей розроблюваної інформаційної системи, які виділяють її серед конкурентів:

1. Інтелектуальна персоналізація клієнтського досвіду: система аналізує поведінку користувачів, накопичує історію їхніх дій та автоматично формує персоналізовані пропозиції, що скорочує час пошуку потрібних товарів.

2. Адаптивна модель ціноутворення: алгоритми аналізують реакцію споживачів на зміну вартості та автоматично коригують ціни для максимізації прибутку і стимулювання продажів залишків.

3. Розділення обчислювальних і транзакційних процесів: ресурсомісткі аналітичні операції виконуються у фоновому режимі, що забезпечує стабільну швидкодію системи навіть при високому навантаженні.

4. Гнучка структура опису товарів: модель даних дозволяє динамічно додавати специфічні характеристики традиційного одягу без зміни структури бази даних.

5. Надійність та стабільність роботи: серверна генерація контенту забезпечує швидке завантаження сторінок, передбачувану роботу інтерфейсу та зниження залежності від потужності пристроїв користувачів.

1.3 Постановка завдання

Загальний опис завдання: розробити вебплатформу інтернет-магазину традиційного українського одягу, яка інтегрує інтелектуальні модулі персоналізації товарних пропозицій та алгоритм динамічної оптимізації вартості товарів.

Вхідні дані:

– каталог автентичних товарів: кожна одиниця товару містить розширений набір атрибутів (колір, тканина, орнамент, тощо), а також дані про поточні складські залишки;

– профілі та активність користувачів реєстраційні дані клієнтів (ПІБ, телефон, email), історія їхніх взаємодій із сайтом (переглянуті товари, додавання до кошика, історія завершених замовлень);

– базова собівартість одиниці виробу та цільова маржинальність для розрахунку початкової ціни.

Вихідні дані:

– сформовані для кожного клієнта добірки «Рекомендовано для вас», засновані на зіставленні його вподобань із характеристиками товарів;

– актуалізовані ціни на товари, перераховані алгоритмом відповідно до ринкової ситуації;

– статистичні дані для адміністратора щодо зміни попиту, залишків на складі та ефективності роботи рекомендаційних алгоритмів.

Рекомендаційна система: працює за принципом знаходження подібностей між користувачами та товарами. Якщо клієнт цікавиться вишиванками з геометричним орнаментом певного регіону, система автоматично підбирає аналоги або аксесуари з подібними характеристиками, знижуючи когнітивне навантаження при виборі.

Динамічне ціноутворення: реалізує стратегію максимізації прибутку та оборотності. Для товарів із високим попитом (багато переглядів/швидкий продаж) ціна автоматично коригується в бік збільшення маржі. Для товарів,

що мають низьку оборотність (залежалися на складі), ціна знижується для прискорення вивільнення капіталу та складського простору.

Постановка завдання:

1. Проаналізувати предметну область та спроєктувати реляційну базу даних, що підтримує зберігання специфічних метаданих етно-виробів та історію цінових змін.

2. Розробити програмне ядро вебсайту, реалізувавши повний цикл електронної комерції (каталог, фільтрація, кошик, замовлення).

3. Розробити та впровадити алгоритм колаборативної фільтрації для реалізації модуля персоналізованих рекомендацій.

4. Розробити алгоритм динамічного ціноутворення на основі регресійного аналізу даних про попит та складську оборотність.

5. Створити адміністративний інтерфейс для моніторингу залишків та ручного коригування параметрів алгоритмів.

Очікувані результати: функціонуюча вебплатформа, яка за рахунок автоматизації аналітичних процесів мінімізує вплив людського фактора, підвищує середній чек замовлення через персоналізацію та оптимізує прибутковість магазину за допомогою гнучкого управління ціною.

Висновки до розділу 1

У ході аналізу предметної області встановлено, що існуючі рішення в сегменті етно-одягу обмежені ручним керуванням і не враховують специфічні характеристики товарів.

Сформована постановка завдання визначає етапи проєктування бази даних, розроблення програмних модулів персоналізації та алгоритмів динамічного ціноутворення. Реалізація проєкту дозволить мінімізувати вплив людського чинника та підвищити ефективність взаємодії з користувачем за рахунок адаптації пропозицій до його індивідуальних запитів.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

2.1. Моделювання поведінки продукту

Проектування програмного забезпечення, зокрема інтернет-магазину традиційного одягу, вимагає чіткого формального опису процесів взаємодії між користувачем та системою. Для повноцінного представлення потрібен перелік функціональних і нефункціональних вимог.

Функціональні вимоги

Система реєстрації та авентифікації:

– користувач реєструється на сайті із використанням особистих даних (прізвище та ім’я, електронна пошта);

– при реєстрації система верифікує користувача, надсилаючи лист з підтвердженням на введену електронну адресу;

– у разі втрати паролю користувач може відновити його за допомогою отримання листа з відновленням на зареєстровану електронну пошту.

Каталог товарів:

– перегляд товарів магазину;

– фільтрація каталогу за категорією та критеріями (EAV-патерн).

Динамічне ціноутворення: автоматичне коригування цін на снові аналізу попиту на товар, за допомогою *назва алгоритму*.

Персоналізовані рекомендації:

– система збирає дані про активність користувачів і логує їх у базі даних;

– фоновий обробник періодично збирає останні активності з бази даних і формує рекомендації для поточних користувачів або повертає їм найпопулярніші товари, якщо замало діяльності для формування особистих рекомендацій;

– розрахунок відбувається за допомогою методу k-НС.

Кошик та замовлення:

– користувач, переглядаючи товари з каталогу, додає або видаляє товари в кошику;

– оформлюючи замовлення клієнт надає дані про отримувача, обирає спосіб оплати та доставки;

– клієнту доступний статус виконання замовлення.

Нефункціональні вимоги

Продуктивність:

– швидкий відгук інтерфейсу (до 2с);

– розподіл навантаження на фонові процеси для пришвидшення обробки даних.

Масштабованість: створення гнучкої системи з модулів, що можна замінити або модифікувати без суттєвих виправлень у ядрі системи.

Безпека: персональні дані користувача та магазину захищені від зовнішнього втручання.

Відмовостійкість та незалежність сутностей: при відмові роботи обробки рекомендацій та оптимізації цін, сайт працюватиме у штатному режимі і навпаки, якщо відмовить сайт, то сервер для фонової обробки не зупиниться.

Діаграма діяльності (рис. 2.1) інтернет-магазину традиційного одягу «Вишиванка Ґаздиня» ілюструє процес придбання, який передбачає взаємодію між двома ключовими учасниками: покупцем та системою.

Клієнт:

– реєструється/авторизується в системі для доступу до персоналізованих пропозицій;

– переглядає каталог товарів, використовуючи фільтри за категоріями;

– додає обрані позиції до віртуального кошика;

– вводить дані для доставки (адреса, контактна інформація);

– обирає метод оплати та підтверджує замовлення.

Система:

– забезпечує відображення актуальних цін та наявності товарів у реальному часі;

– автоматично розраховує підсумкову вартість замовлення з урахуванням знижок та вартості доставки;

– фіксує замовлення в базі даних та генерує унікальний номер транзакції;

– надсилає сповіщення про статус замовлення покупцю (лист на електронну пошту або сповіщення в месенджері чи SMS).

Послідовність процесу придбання:

1. Покупець переходить до каталогу та обирає товар.
2. Система ініціює запит до бази даних та відображає картку товару.
3. Покупець додає товар до кошика.
4. Після формування повного списку покупок, користувач переходить до оформлення замовлення (Checkout).
5. Система перевіряє коректність введених даних та наявність товару на складі.
6. Покупець підтверджує замовлення.
7. Система оновлює стан замовлення в БД та надсилає підтвердження.

Рисунок 2.1 - Діаграма варіантів використання (Use Case Diagram) системи. Джерело: розроблено автором

На рисунку 2.1 наведено діаграму варіантів використання (Use Case Diagram), яка відображає основні функціональні вимоги до системи інтернет-магазину. Виділено двох основних акторів: «Клієнт» та «Адміністратор».

Основна увага зосереджена на реалізації інтелектуальних механізмів системи: прецеденти «Перегляд персональних рекомендацій» (для користувачів) та «Налаштування правил ціноутворення» (для адміністратора). Процес доступу до системи та управління контентом реалізовано через механізм обов’язкової аутентифікації, що забезпечує дотримання політики контролю доступу на основі ролей (RBAC - Role-Based Access Control). Для нових користувачів система передбачає використання алгоритму рекомендацій, що базується на популярності товарів (вирішення проблеми «холодного старту»).

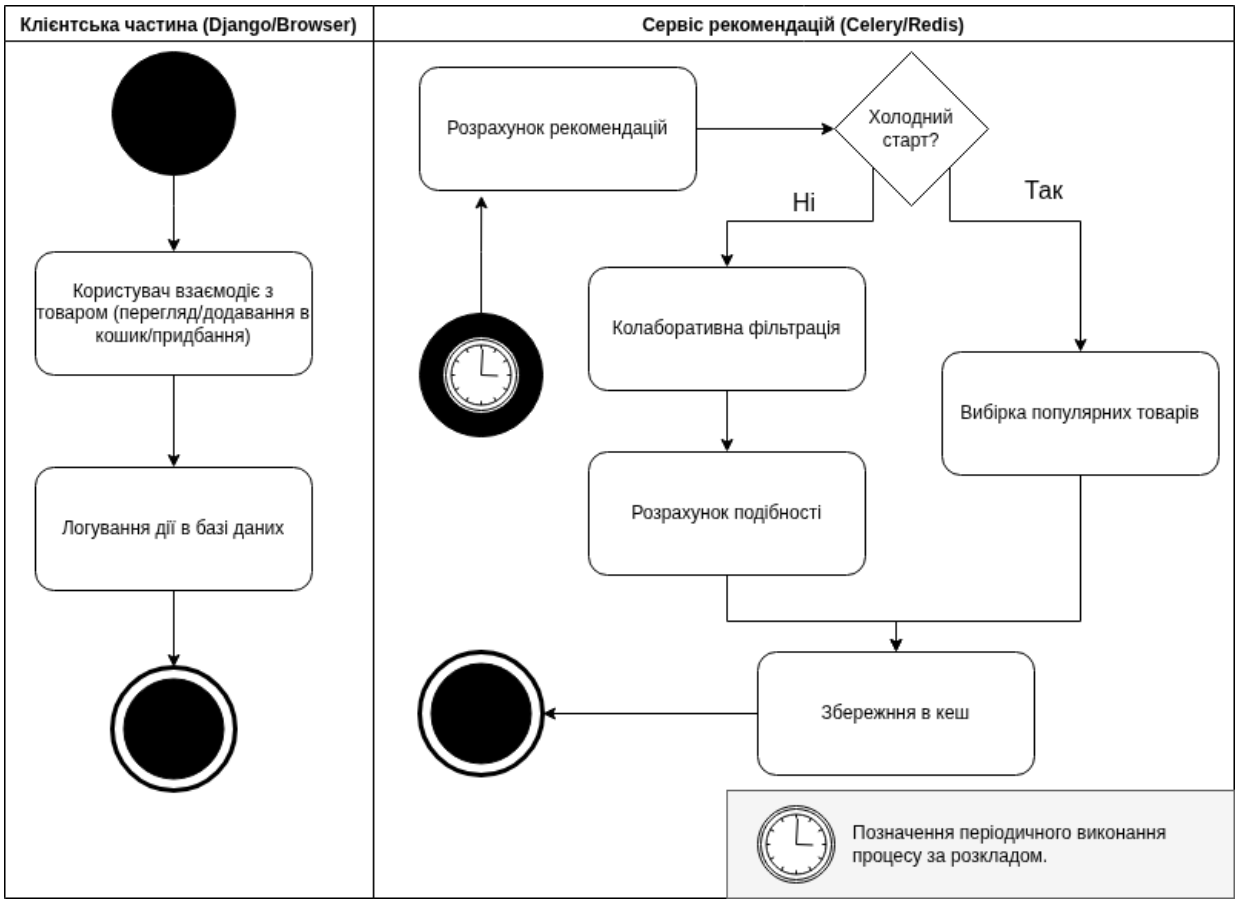


Рисунок 2.2 – Діаграма діяльності процесу формування рекомендацій.

Джерело: розроблено автором.

Окрім транзакційного процесу, система реалізує фонову обробку даних для персоналізації пропозицій. Даний процес описує взаємодію між користувачем та фоновим сервісом системи.

Користувач:

– здійснює навігацію сайтом (перегляди, кліки, додавання до кошика);

– формує історію взаємодій, яка накопичується в журналі подій системи.

Фоновий сервіс (Система):

– збирає дані про поведінку користувача за певний часовий проміжок (лог подій);

– застосовує алгоритм K-найближчих сусідів (k-НС) для аналізу подібності запитів;

– оновлює кеш рекомендацій для відображення на головній сторінці або в особистому кабінеті;

– виконує перерахунок динамічних цін на основі коефіцієнта попиту.

Послідовність формування рекомендацій:

1. Користувач здійснює взаємодію з об’єктами системи (товарами).
2. Подія взаємодії фіксується в базі даних.
3. Система активує фоновий воркер (Celery) для обробки черги задач.
4. Воркер отримує дані з логів та запускає алгоритм рекомендацій.
5. Результати (ID рекомендованих товарів) записуються до кеш-сховища (Redis) або таблиці рекомендацій.
6. При наступному завантаженні сторінки система відображає підготовлені пропозиції користувачу.

Логіка роботи побудована на принципі асинхронності: при виникненні події (перегляд або купівля товару), інформація фіксується у базі даних (лог подій). Розрахунок рекомендацій не відбувається синхронно з дією користувача, що дозволяє уникнути затримок під час HTTP-запитів. Фонова задача, що ініціюється планувальником за розкладом, аналізує накопичені дані, виконує алгоритм колаборативної фільтрації (або вибірку популярних товарів у разі «холодного старту») та оновлює кеш для клієнта. Такий підхід забезпечує стабільність системи та її масштабованість.

2.2. Моделювання архітектури та структури продукту

Архітектура системи базується на принципах модульності та відокремлення бізнес-логіки від представлення даних. Для реалізації серверної частини обрано фреймворк Django, що використовує модифікований

архітектурний шаблон MVT (Model-View-Template). Цей вибір дозволяє чітко розділити ключові функції системи, забезпечуючи її високу масштабованість:

Model (Модель): відповідає за персистентність даних та бізнес-логіку. Сюди включено логіку ціноутворення, управління товарними залишками та взаємодію з інтелектуальними модулями.

View (Представлення): отримує HTTP-запити, викликає необхідну бізнес-логіку та передає результат до шаблонізатора.

Template (Шаблон): відображає кінцевий інтерфейс користувача (фронтенд).

Таке відокремлення є критично важливим, оскільки дозволяє масштабувати бізнес-логіку, яка реалізується у фонових сервісах Celery/Redis, незалежно від фронтенду. Це запобігає блокуванню основного потоку запитів під час виконання ресурсоємних обчислень, таких як перерахунок динамічних цін або формування рекомендацій**.**

На основі розроблених функціональних вимог побудовано UML-діаграму класів (Class Diagram), яка відображає логічну структуру взаємодії компонентів, їхні внутрішні атрибути, методи та типи зв'язків (рис. 2.3).

Спроєктована об'єктна модель декомпонована на три ключові архітектурні шари: шар представлення (шаблони та контролери веб-інтерфейсу), сервісний шар (ізольована бізнес-логіка) та шар об'єктно-реляційного відображення (ORM-моделі Django).

Центральним елементом інтеграції бізнес-правил у процесі оформлення замовлень є використання патерну “Сервісний шар”, представленого класом OrderService. Винесення логіки координації транзакцій з контролерів представлень у виділений сервісний клас дозволяє уникнути перевантаження views та забезпечує повторне використання коду.

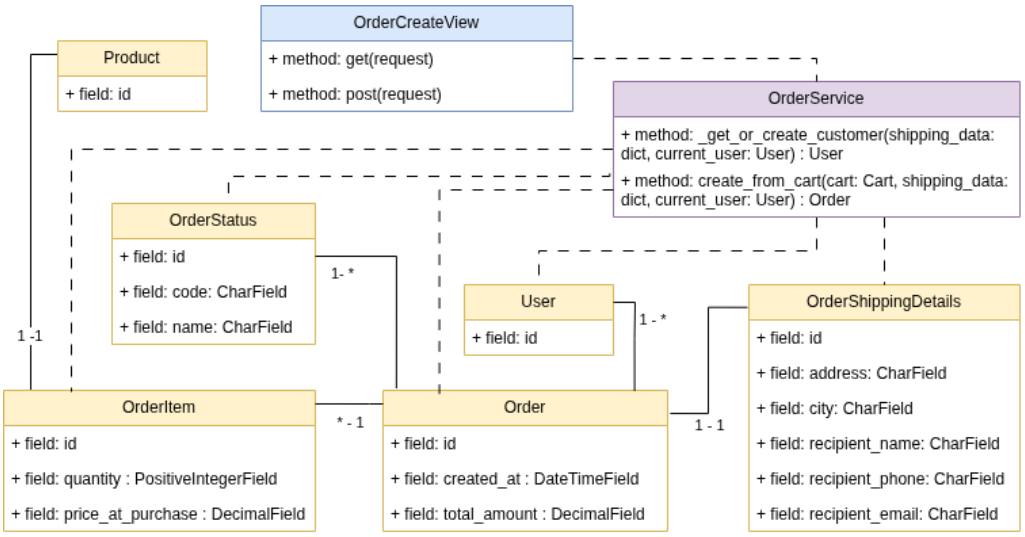


Рисунок 2.3 - Діаграма класів.

Джерело: розроблено автором

На спроєктованій діаграмі (рис. 2.3) зафіксовано такі класи та логіку їхньої взаємодії:

1. OrderCreateView — клас представлення на основі вбудованих засобів вебфреймворку, що відповідає за обробку HTTP-запитів клієнта (get(), post()). Він ініціалізує об'єкт сесійного кошика Cart, валідує вхідну HTML-форму і, у разі її успішності, делегує виконання транзакції сервісному шару через виклик методу create_from_cart().

2. OrderService — службовий інтерфейсний клас, що оркеструє процесами створення сутностей всередині єдиної атомарної транзакції баз даних. Його статичний метод _get_or_create_customer() інкапсулює логіку автоматичної реєстрації анонімних користувачів, які оформлюють гостьове замовлення, динамічно генеруючи для них обліковий запис у моделі User. Метод create_from_cart() забезпечує зчитування даних із кошика, послідовну генерацію екземплярів моделей та їх масове збереження.

3. Шар ORM-моделей (сутності збереження даних). Спроєктована структура атрибутів класів використовує рідні об'єктні типи фреймворку Django (CharField, DateTimeField, DecimalField, PositiveIntegerField), що дозволяє абстрагуватися від низькорівневих типів конкретної СУБД,

реалізувати вбудовані механізми захисту від SQL-ін'єкцій та забезпечити автоматичну генерацію міграцій.

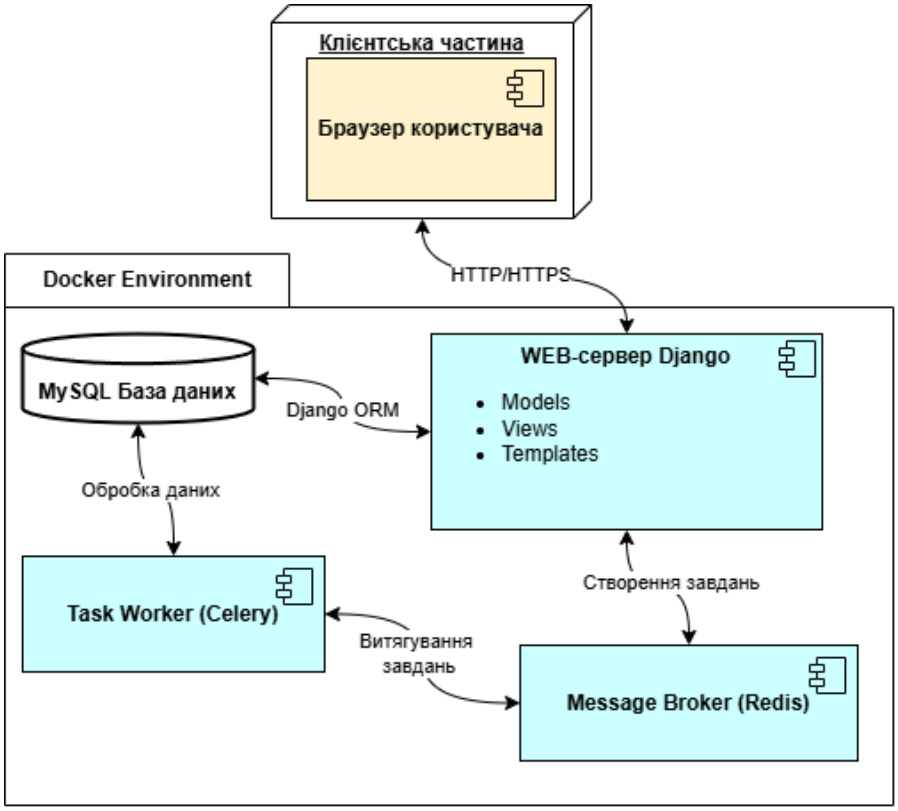


Рисунок 2.4 - Діаграма компонентів (Component Diagram)

Джерело: розроблено автором

На високорівневому рівні (рис. 2.4) система спроєктована як сукупність взаємопов'язаних сервісів, що функціонують у контейнерах Docker:

– клієнтський рівень: представлений браузером користувача, який ініціює запити до Web-сервісу;

– Web-сервіс (Django): основний рівень бізнес-логіки, який обробляє транзакційну логіку (кошик, замовлення);

– сервер баз даних (MySQL): забезпечує персистентне зберігання структурованих даних (товари, користувачі, лог взаємодій);

– асинхронний рівень (Celery + Redis): сервіс асинхронних задач, що виконує ресурсоємні обчислення, не блокуючи Web-сервіс.

Ця багатошарова архітектура забезпечує відмовостійкість та незалежність сутностей: при відмові роботи інтелектуальних модулів (Celery/Redis), Web-сервіс продовжує працювати у штатному режимі.

База даних системи спроєктована з використанням реляційної моделі (СУБД MySQL), що забезпечує високу цілісність даних (ACID-compliance). Схема БД приведена до третьої нормальної форми (3NF), що мінімізує надмірність даних та виключає аномалії при оновленні інформації (на рис. 2.5 представлено ERD).

Ключові сутності

Таблиця користувачів: фундамент системи, ідентифікує суб'єктів, забезпечує доступ до персоналізованих функцій. Вона пов'язана з таблицею замовлень та логами взаємодій (перегляд, доадвання в кошик, придбання) для аналізу та рекомендацій.

Таблиця товарів: центральний елемент каталогу, що містить як базову, так і актуальну ціну, яка динамічно коригується. Взаємодіє з ієрархічною структурою категорій та допоміжними таблицями характеристик для відображення детальних параметрів етно-виробів.

Таблиця замовлень: логічно об'єднує клієнта та обрані товари, фіксуючи поточний статус купівлі. Зберігає деталі доставки, а також «знімок» ціни на момент покупки для фінансової цілісності.

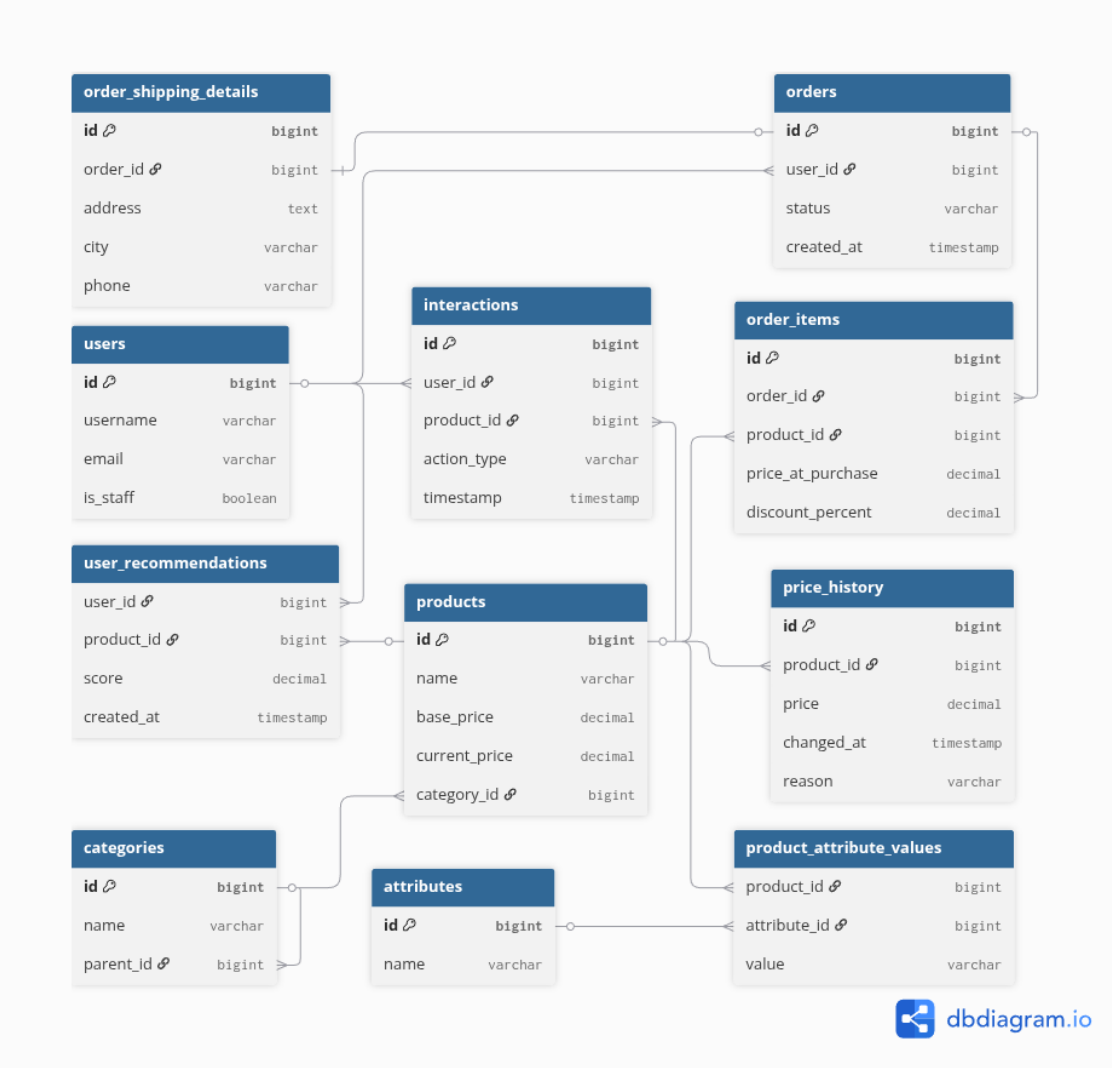


Рисунок 2.5 — ER-діаграма бази даних

Джерело: розроблено автором (створено за допомогою dbdiagram.io [8])

2.3 Опис алгоритмів та математичних моделей

Для інтелектуального формування індивідуальних товарних добірок традиційного одягу було проаналізовано три поширені класи методів: контентну фільтрацію, матричну факторизацію (зокрема, сингулярне розкладання матриць — SVD) та колаборативну фільтрацію на основі пам'яті за методом k-найближчих сусідів.

Контентна фільтрація фокусується на лінгвістичному описі та фіксованих тегах виробів. Такий підхід призводить до ефекту "надлишкової спеціалізації", коли система пропонує покупцеві лише ідентичні моделі одягу,

звужуючи його користувацький досвід та обмежуючи крос-категоріальні продажі.

Для формування рекомендацій використано алгоритм колаборативної фільтрації на основі методу k-найближчих сусідів (k-НС). Він визначає схожість між користувачами шляхом порівняння векторів їхньої активності.

Математична модель: для кількісної оцінки подібності векторів використовується косинусна подібність, математичний вираз якої представлено формулою (2.1):

$$\text{similarity}(A, B) = \cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}} \qquad (2.1),$$

де *A*, *B* - вектори взаємодій двох користувачів;

A_i, B_i - компоненти векторів, що відображають вагу взаємодії користувачів з товаром і;

n - розмірність векторів (загальна кількість товарів у системі).

Застосування в продукті: система формує вектор *A*, де значення A_i - це агрегований рейтинг товару *i* (на основі таблиці *interactions*: перегляд=1, кошик=3, купівля=5). Celery-воркер збирає вектори для всіх користувачів, обчислює подібність до цільового користувача, обирає *K* "найближчих сусідів" (найбільший показник *similarity*) і рекомендує товари, які ці сусіди придбали, а цільовий користувач - ні. Результат зберігається в кеш-таблицю user_recommendations для швидкої видачі користувачу.

Під час проєктування аналітичного модуля динамічного коригування цін розглядалися три підходи: прогнозування на основі штучних нейронних мереж (рекуррентних LSTM або градієнтного бустингу), евристичні правила (статичні знижки менеджерів) та економіко-математична модель цінової еластичності попиту (Price Elasticity of Demand, PED).

Нейромережеві моделі та алгоритми машинного навчання, попри їхній високий потенціал, є математичною "чорною скринькою". Вони вимагають накопичення багаторічних ретроспективних часових рядів і схильні до перенавчання (overfitting) за умов різкої зміни макроекономічних факторів, що робить їх нестабільними для малого та середнього e-commerce бізнесу.

Евристичні статичні правила (наприклад, сезонні знижки), що широко застосовуються в e-commerce ритейлі, повністю залежать від людського фактора, не мають під собою об'єктивного математичного підґрунтя і призводять до недоотримання маржинального прибутку підприємства.

Для реалізації динамічного ціноутворення використано модель цінової еластичності попиту, що базується на класичних положеннях математичної економіки, адаптованих до цифрового ринку. Вона дозволяє алгоритмічно оцінити чутливість споживачів до коливань цін у режимі реального часу, автоматично знижуючи ціну для нееластичних залишків на складах з метою стимулювання збуту та підвищуючи її на високоліквідні товари для максимізації доходу, що забезпечує прогнозовану та відмовостійку роботу системи фонових обробників.

Математично коефіцієнт еластичності E_d розраховується як відношення відсоткової зміни обсягу попиту Q до відсоткової зміни ціни P [10]:

$$\mathrm{E_d} = \frac{\Delta \mathrm{Q}/\mathrm{Q}}{\Delta \mathrm{P}/\mathrm{P}} = \frac{\%\Delta Q}{\%\Delta \mathrm{P}} \quad (2.2),$$

де $\mathrm{E_d}$ -коефіцієнт цінової еластичності попиту;

ΔQ - абсолютна зміна обсягу попиту (кількості проданих товарів);

Q - початковий обсяг попиту;

ΔP абсолютна зміна ціни товару;

P - початкова ціна товару.

Застосування в продукті: алгоритм використовує історичні дані з таблиць *price_history* та *order_items*. Система аналізує, як зміна ціни P в минулі періоди вплинула на продажі Q.

Якщо (попит нееластичний), ціна може бути підвищена на коефіцієнт *k*, оскільки покупці менш чутливі до зміни вартості.

Якщо $|E_d| < 1$ (попит еластичний), система рекомендує зниження ціни для нарощування обсягу продажів. Розрахунок виконується фоново раз на добу, а запропоновані зміни цін записуються в історію, після чого менеджер або автоматика (залежно від налаштувань) можуть застосувати їх до *current_price* в таблиці *products*.

Висновки до розділу 2

Проєктування програмного продукту забезпечило формалізацію вимог, архітектури та алгоритмічної основи системи.

Моделювання поведінки визначило ключові функціональні вимоги, включаючи роботу каталогу з фільтрацією за патерном EAV, кошик, авторизацію, а також реалізацію інтелектуальних модулів динамічного ціноутворення та персоналізованих рекомендацій. Була розроблена логіка асинхронного процесу: взаємодії користувача фіксуються в логах, а фоновий воркер Celery періодично обробляє дані, використовуючи алгоритми, для оновлення кешу.

Архітектура та структура побудована на багатошаровому принципі з використанням фреймворку Django та Docker-контейнерів. Це забезпечить незалежність сутностей та відмовостійкість, розділяючи Web-сервіс, сервер баз даних та асинхронний рівень (Celery + Redis) для ресурсоємних обчислень. Реляційна база даних спроєктована в третій нормальній формі (3NF) для мінімізації надмірності та забезпечення цілісності даних.

Опис алгоритмів закріпив математичні моделі для інтелектуальних модулів:

Персоналізовані рекомендації: Використовується метод колаборативної фільтрації на основі k-найближчих сусідів (k-НС) з косинусною подібністю для вимірювання схожості між векторами взаємодій користувачів.

Динамічне ціноутворення: застосовано модель цінової еластичності попиту (PED). Це дозволяє автоматично підвищувати ціну при нееластичному попиті ($|E_d| < 1$) для максимізації маржі та знижувати при еластичному попиті ($|E_d| > 1$) для збільшення обсягу продажів.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1. Реалізація та констуювання програмного продукту

Процес конструювання та розробки вебсайту інтернет-магазину традиційного одягу вимагав створення гнучкої, відмовостійкої та слабкопов’язаної архітектури. Це зумовлено необхідністю поєдання стандартних транзакційних операцій електронної комерції з високонавантаженими обчислювальними процесами інтелектуальних модулів (алгоритмів КНС та цінової еластичності). На основі проведеного аналізу системних вимог було сформовано та обґрунтовано технологічний стек.

Мовою програмування обрано Python 3.12. Вибір зумовлений її домінуючим становищем у галузі аналізу даних та розробки штучного інтелекту. Наявність оптимізованих математичних бібліотек дозволяє швидко реалізовувати матричні обчислення, а лаконічний синтаксис прискорює написання та підтримку бізнес-логіки.

Для веброзробки обрано Python вебфреймворк Django 5.0. Обраний як високорівневий фреймворк для швидкої та безпечної розробки. Його вбудована підсистема автентифікації (django.contrib.auth), механізм сесій, та доволі потужний інструмент ORM (Object-Relational Mappting - Об’єктно-реляційне відображення) дозволяють абстрагуватися від прямих SQL-запитів і забезпечують високу швидкість обробки даних каталогу з пагінацією та фільтрацією.

Для збереження даних обрано СУБД MySQL 8.0, вона є необхідною для забезпечення ACID-сумісності транзакцій (оформлення замовлень, списання залишків). MySQL обрано через її високу продуктивність при операціях читання-запису, надійність та повну сумісність із Django ORM.

Брокер повідомлень Redis та черга задач Celery. Використання цієї зв'язки є ключовим інженерним рішенням проєкту. Оскільки розрахунки

косинусної подібності для k-НС-рекомендацій та інтегральні обчислення коефіцієнтів еластичності PED потребують значних обчислювальних ресурсів, їх виконання в основному синхронному потоці HTTP-запитів призвело б до блокування вебсервера. Redis виступає як швидке сховище в оперативній пам'яті для буферизації задач, а Celery функціонує як ізольований фоновий воркер, що виконує обчислення асинхронно.

Бібліотека наукових обчислень NumPy застосована як основний інструмент для оптимізації обробки матриць взаємодій та математичних розрахунків в аналітичних воркерах. Використання базових засобів мови Python (вкладених циклів for та стандартних списків list) спочатку не мало б суттєвої ваги для проведення розрахунків, проте за діяльності кількох десятків тисяч користувачів на сайті призвело б до критичних витрат оперативної пам'яті та процесорного часу через динамічну типізацію та надбудови інтерпретатора над кожним об'єктом. Матричне подання даних за допомогою об'єктів numpy.ndarray забезпечує виконання операцій на рівні скомпільованого С-коду. Завдяки механізму векторалізації алгоритми оперують цілими масивами даних одночасно, без явного використання циклів, що кардинально знижує часову складність алгоритму та підвищує пропускну здатність фонового шару.

Середовище розробки та платформа контейнеризації. Розробка здійснювалася в інтегрованому середовищі PyCharm 2026.1.1. Для ізоляції залежностей та забезпечення ідентичності конфігурації на етапах розробки й розгортання застосовано платформу контейнеризації Docker та інструмент оркестровки Docker Compose.

Для забезпечення чистоти кодової бази та дотримання принципу розділення відповідальності, архітектура Django-проєкту декомпонована на чотири автономні програмні модулі (apps).

Модуль автентифікації та профілів (users). Центральним елементом є кастомізована модель користувача User, що успадковує властивості класу

AbstractUser і явно відображається на реляційну таблицю users СУБД MySQL. Щоб уникнути конфліктів зв'язків у системі міграцій Django, для полів groups та user_permissions визначено унікальні ідентифікатори зворотного зв'язку (related_name). Для реєстрації розроблено спеціалізовану форму CustomUserCreationForm, а інтерфейсна логіка реалізована за допомогою представлень на основі класів (CBV) RegisterView, який автоматично здійснює авторизацію клієнта в сесії за допомогою методу login() після успішної валідації форми, та CustomLoginView.

Модуль керування асортиментом та інтерфейсом каталогу (catalog). Даний застосунок містить сутності для зберігання структури товарів: Category (ієрархічні категорії), Product (базова інформація про одяг), а також класи Attribute та ProductAttributeValue, що реалізують патерн проєктування EAV (Entity-Attribute-Value) для гнучкого опису різнорідних характеристик моделей традиційного одягу (матеріал, розмір, орнамент) без модифікації схеми БД. Управління відображенням контенту покладено на представлення HomeView (формує контекст головної сторінки з акційними пропозиціями та персоналізованими картками товарів) та ProductListView, яке за допомогою Django ORM реалізує складні механізми багатокритеріальної фільтрації та сортування.

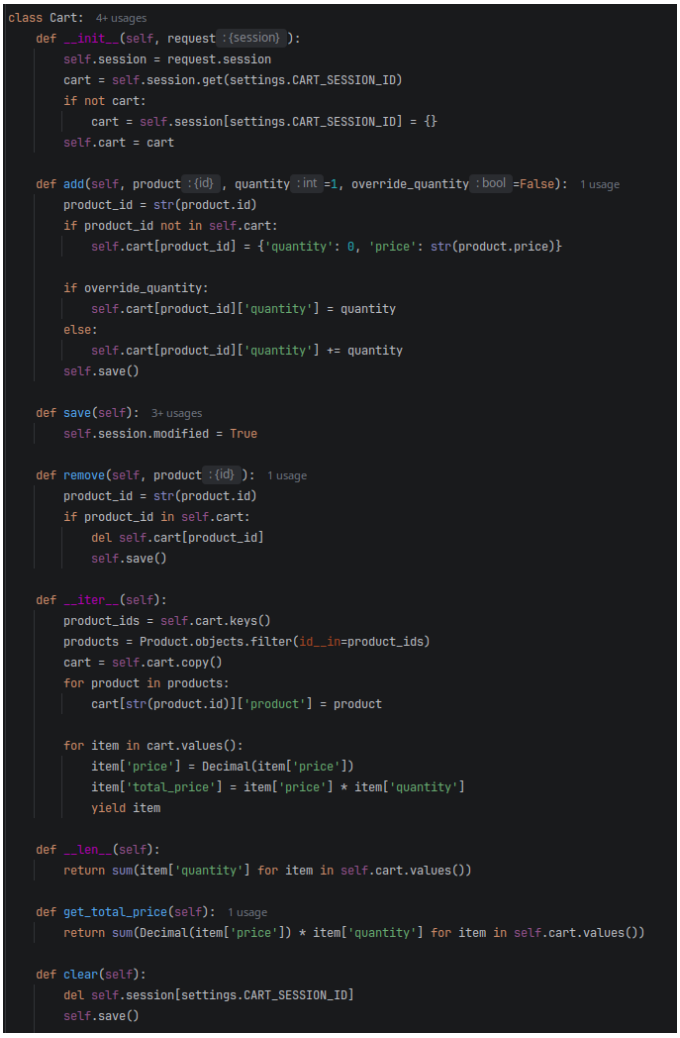
Транзакційний модуль обробки замовлень (orders). Керує процесом накопичення обраних позицій, логістичними даними та фіксацією фінансових угод через сутності Order, OrderItem та OrderShippingDetails. Для забезпечення швидкодії системи, кошик покупця реалізовано як об'єктний шар у межах поточних сесій користувача без постійного звернення до дискової підсистеми бази даних. Взаємодія з кошиком здійснюється через контролери представлень cart_add (із декоратором @require_POST), cart_detail та cart_remove.

Аналітичний модуль інтелектуальної обробки (analytics). Виступає обчислювальним ядром системи. Він ізолює в собі математичні моделі та сутності збереження результатів обчислень: Interaction (сирі логи дій

користувачів), PriceHistory (ретроспективні дані коливання вартості) та UserRecommendation (персоналізовані рейтинги товарів). Також у цьому застосунку розміщено описи асинхронних завдань (tasks) для розподілених воркерів Celery.

Для абстрагування логіки роботи з тимчасовим сховищем товарів розроблено службовий клас Cart (рис. 3.1). Кошик ініціалізується ідентифікатором сесії, що зчитується з глобальних конфігурацій сайту (settings.CART_SESSION_ID = 'cart').

Усі грошові розрахунки всередині класу виконуються з обов'язковим кастингом типів даних у клас Decimal стандартного пакета Python, що гарантує збереження точності округлення і сумісність із типами полів СУБД.



```
class Cart:  4+ usages
    def __init__(self, request :{session} ):
        self.session = request.session
        cart = self.session.get(settings.CART_SESSION_ID)
        if not cart:
            cart = self.session[settings.CART_SESSION_ID] = {}
        self.cart = cart

    def add(self, product :{id} , quantity :int =1, override_quantity :bool =False):  1 usage
        product_id = str(product.id)
        if product_id not in self.cart:
            self.cart[product_id] = {'quantity': 0, 'price': str(product.price)}

        if override_quantity:
            self.cart[product_id]['quantity'] = quantity
        else:
            self.cart[product_id]['quantity'] += quantity
        self.save()

    def save(self):  3+ usages
        self.session.modified = True

    def remove(self, product :{id} ):  1 usage
        product_id = str(product.id)
        if product_id in self.cart:
            del self.cart[product_id]
            self.save()

    def __iter__(self):
        product_ids = self.cart.keys()
        products = Product.objects.filter(id__in=product_ids)
        cart = self.cart.copy()
        for product in products:
            cart[str(product.id)]['product'] = product

        for item in cart.values():
            item['price'] = Decimal(item['price'])
            item['total_price'] = item['price'] * item['quantity']
            yield item

    def __len__(self):
        return sum(item['quantity'] for item in self.cart.values())

    def get_total_price(self):  1 usage
        return sum(Decimal(item['price']) * item['quantity'] for item in self.cart.values())

    def clear(self):
        del self.session[settings.CART_SESSION_ID]
        self.save()
```

Рисунок 3.1 - знімок екрану із відображенням програмного класу Cart.

Джерело: розроблено автором

Підключення до реляційної СУБД MySQL 8.0 реалізовано за допомогою системних налаштувань проєкту, де конфіденційні дані доступу динамічно імпортуються з файлу оточення за допомогою методу os.getenv(). Для зберігання грошових показників вартості товарів і замовлень у базі даних архітектурно закріплено тип поля Decimal із фіксованою точністю - DecimalField(max_digits=10, decimal_places=2). Це запобігає виникненню похибок округлення floating-point чисел, що є критично важливим для електронної комерції.

Оскільки система розгортається у багатоконтейнерному середовищі Docker Compose [], виникає ризик десинхронізації запусків сервісів. Якщо вебсервер Django або воркер Celery розпочнуть процедуру старту до того, як СУБД MySQL або брокер Redis перейдуть у стан повної готовності приймати мережеві з'єднання, відбудеться аварійне завершення (crash) основних процесів додатка.

Для нівелювання цього ризику на рівні оркестрації Docker Compose було впроваджено механізм моніторингу працездатності (Healthcheck). Контейнер бази даних безперервно виконує внутрішню діагностичну утиліту mysqladmin ping з інтервалом у 3 секунди. Основні сервіси (web, worker) запускаються з умовою condition: service_healthy:

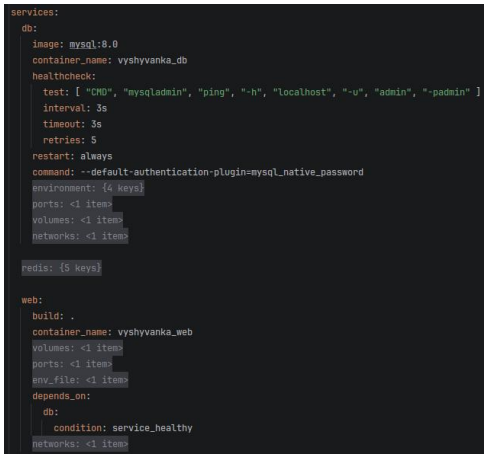


Рисунок 3.2 - знімок екрану із відображенням блоків сервісів з Docker Compose із реалізацією механізму healthcheck.

Джерело: розроблено автором

Асинхронна архітектура обробки даних налаштована через брокер повідомлень Redis за допомогою параметрів CELERY_BROKER_URL. Транспортний протокол обміну повідомленнями використовує серіалізацію у форматі JSON, що оптимізує швидкість пакування та розпакування об'єктів черги завдань воркерами.

Логіка збору сирих даних взаємодій користувачів для аналітичного шару інтегрована безпосередньо у Request-Response життєвий цикл Django представлень (Views). Під час виконання HTTP-запиту клієнта до сторінки конкретного товару, контролер представлення на етапі формування відповіді ініціює асинхронний запис у таблицю interactions через Django ORM, фіксуючи тип дії (view). Операції додавання товару в кошик або переходу на сторінку оплати логуються постфактум, після успішного виконання відповідних POST-методів, що мінімізує мережеве навантаження на систему.

Запуск Celery-воркерів для проведення ресурсомістких математичних обчислень (k-НС та PED) у промислових умовах налаштовується за допомогою планувальника Celery Beat. Враховуючи динаміку оновлення асортименту інтернет-магазину та інтенсивність накопичення кліків, запуск задач оптимізовано для виконання декілька разів на добу за циклічним розкладом (наприклад, кожні 4 або 6 годин у періоди найменшої активності покупців). Це дозволяє підтримувати матрицю персональних рекомендацій та поточні ціни в актуальному стані, не знижуючи швидкість відгуку вебкаталогу.

Для оптимізації рендерингу та зниження навантаження на мережеві канали, представлення ProductListView використовує вбудований механізм пагінації Django, обмежуючи виведення до 12 карток товарів на одній сторінці (paginate_by = 12). Проте при поєднанні пагінації з багатокритеріальною фільтрацією за ціною та категоріями виникає проблема втрати активних фільтрів при переході на наступну сторінку, оскільки стандартні посилання пагінатора перезаписують GET-параметри URL.

Для розв'язання цієї проблеми в метод get_context_data було інтегровано алгоритм динамічного кодування рядка запиту. Програма копіює поточний словник GET-параметрів (self.request.GET.copy()), вилучає з нього ключ page (щоб уникнути дублювання ітераторів сторінок) та за допомогою методу urlencode() перетворює словник фільтрів на уніфікований рядок типу category=men&min_price=500.

Цей кодований рядок передається в контекст шаблону як змінна query_string і динамічно підставляється в HTML-посилання навігації. Це дозволяє зберігати всі вибрані фільтри за ціною чи категорією незмінними при кліках по номерах сторінок.

Сховище даних реалізовано на реляційній СУБД MySQL з використанням схеми, приведеної до третьої нормальної форми (3NF). Це забезпечує мінімізацію надмірності даних та їхню цілісність. Загальна структура зв’язків між сутностями наведена на ER-діаграмі (рис. 2.4).

Таблиця товарів (Products): Центральний елемент каталогу, що містить як базову, так і актуальну ціну (current_price), яка динамічно коригується. Вона взаємодіє з допоміжними таблицями характеристик, що реалізовані за патерном EAV (Entity-Attribute-Value), для зберігання специфічних метаданих етно-виробів (техніка вишивки, регіональний орнамент).

Таблиця користувачів (Users): Ідентифікує суб'єктів системи, забезпечуючи доступ до персоналізованих функцій.

Таблиця замовлень (Orders) та Деталі замовлень (OrderItems): Фіксує загальний статус купівлі. Зберігає «знімок» ціни на момент транзакції для фінансової цілісності.

Логи взаємодій (Interaction Log): Фіксує поведінку користувачів (перегляди=1, кошик=3, купівля=5), що є основою для Celery-воркера для запуску алгоритму колаборативної фільтрації k-НС.

Історія цін (Price History): Зберігає історичні зміни цін. Використовується алгоритмом Цінової еластичності попиту (PED) для аналізу впливу ціни на обсяги продажів.

3.2. Тестування програмного продукту

Процес тестування розробленого вебсайту інтернет-магазину традиційного одягу спрямований на верифікацію відповідності реалізованого MVP-прототипу поставленим технічним, логічним та функціональним вимогам. Зважаючи на динамічний характер розробки та специфіку архітектури, основний акцент було зроблено на ручному функціональному тестуванні (Manual Functional Testing) методом «чорної скриньки» (Black-Box Testing) за заздалегідь визначеними користувацькими сценаріями, а також на інтеграційній верифікації взаємодії розподілених компонентів системи.

Функціональне тестування проводилося з метою перевірки того, чи виконує кожен компонент інтерфейсу користувача (UI) заявлені бізнес-вимоги без аналізу внутрішньої структури коду. Для цього було спроєктовано масив тестових сценаріїв (Test Cases), які охоплюють повний цикл взаємодії покупця з платформою: від реєстрації до моменту формування замовлення та перерахунку цін.

Результати проведення функціонального тестування та порівняння очікуваних результатів із фактично отриманими зведено в таблицю 3.3.

Таблиця 3.1 - функціональне тестування сайту.

Об'єкт перевірки	*Вхідні умови / Дія*	*Очікуваний результат*	*Фактичний результат*	*Статус*
Модуль users (Реєстрація)	Введення унікальних username, email та валідного пароля	Створення запису в таблиці users, автоматичний вхід у систему, редирект на home	Користувач створений, сесія активна, відбувся перехід на головну сторінку	Успішно
Модуль catalog (Фільтрація)	Вибір категорії в меню та встановлення діапазону цін	Зміна GET-параметрів URL, виведення товарів лише заданого типу та вартості	Товари відфільтровано, невідповідні позиції приховано з екрана	Успішно
Модуль catalog (Пагінація)	Перехід на сторінку 2 за наявності активних фільтрів	Перехід на наступний батч товарів із повним збереженням стейту фільтрації в URL	Пагінатор згенерував посилання з query_string, фільтри не скинулися	Успішно
Модуль orders (Кошик - додавання)	Натискання кнопки «Додати в кошик» на картці товару	Створення/модифікація словника в request.session, оновлення лічильника	Товар зареєстровано в сесії, лічильник у шапці сайту змінився	Успішно
Модуль orders (Кошик - оновлення)	Зміна кількості або видалення позиції в кошику	Повний динамічний перерахунок total_price або видалення ключа із сесії	Вартість перерахована коректно, видалений товар зник із таблиці кошика	Успішно

Оскільки система є розподіленою і функціонує як сукупність Docker-контейнерів, критично важливим етапом є інтеграційне тестування - перевірка

коректності обміну даними між вебзастосунком Django, реляційною СУБД MySQL, брокером Redis та воркерами Celery.

Верифікація інтеграційної взаємодії здійснювалася шляхом наскрізного моніторингу журналів системних подій (логів) у терміналі розробника. Процес перевірки складався з двох етапів:

Перевірка логів Docker Compose під час запуску показала, що завдяки налаштованим параметрам healthcheck контейнер web очікує повної готовності порту MySQL (service_healthy) і лише після цього ініціює з'єднання. Це повністю усунуло проблему аварійного завершення процесів через мережеві затримки.

Для верифікації працездатності інтелектуальних алгоритмів (k-НС та PED) у терміналі контейнера worker було виконано тестовий виклик задач. Аналіз логів показав, що воркер успішно підключається до Redis, вилучає завдання з черги, виконує обчислення і здійснює запис результатів у MySQL (рис 3.3).

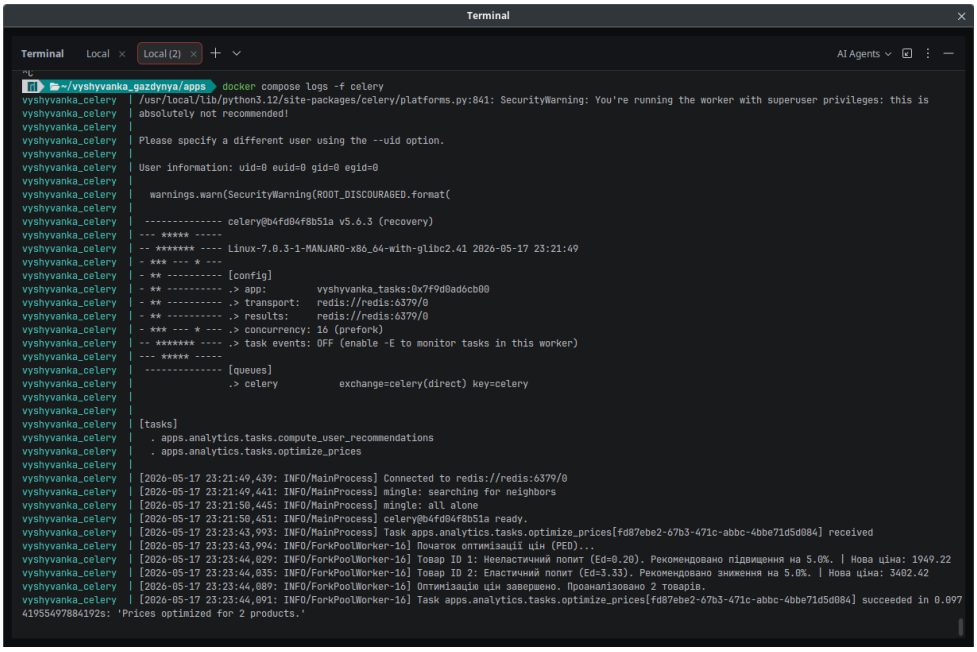


Рисунок 3.3 - Знімок екрану із відображенням логів Celery-воркера під час успішного виконання задач аналітики.

Джерело: розроблено автором

У межах тестування інтелектуального шару окрему увагу було приділено перевірці стійкості алгоритмів до специфічних або «порожніх» станів даних (крайових випадків), які могли б призвести до критичних помилок (Runtime Errors) у промислових умовах:

Тестування проблеми «холодного старту»: імітувався сценарій, коли новий користувач не має жодного запису в таблиці взаємодій interactions. Тестування підтвердило, що архітектурний механізм обробки винятків перехоплює нульовий результат роботи алгоритму k-НС і замість аварійної зупинки сторінки динамічно підтягує в контекст HomeView масив найбільш популярних товарів загального каталогу.

Перевірка стійкості моделі цінової еластичності (PED): для товарів, ціна на які є статичною з моменту внесення в каталог, показник зміни ціни ΔP дорівнює нулю. Ручний запуск розрахунку підтвердив, що конструкція try-except ZeroDivisionError, інтегрована в обробник задачі optimize_prices_by_ped_task, успішно запобігає виникненню помилки ділення на нуль. Система коректно ігнорує переоцінку таких позицій, зберігаючи стабільність фонового процесу.

3.3. Використання програмного продукту

Розроблений програмний продукт є вебсайтом інтернет-магазину традиційного одягу, інтерфейс якого побудовано за принципом Server-Side Rendering (SSR) на основі шаблонізатора Django. Вся взаємодія користувача з платформою, включаючи фільтрацію, пагінацію та керування кошиком, реалізована за допомогою класичних HTTP-запитів (GET та POST) і стандартних HTML5-форм. Такий підхід гарантує високу стабільність інтерфейсу, незалежність від клієнтських JS-скриптів та передбачуваність станів системи.

Нижче наведено інструкцію користувача із описом основних екранних форм та логіки взаємодії з MVP-прототипом.

При первинному переході за адресою розгортання системи користувач потрапляє на Головну сторінку. Сторінка містить верхню навігаційну панель (Header) із посиланнями на каталог, кошик та модулі авторизації, а також центральний банер із акційними пропозиціями.

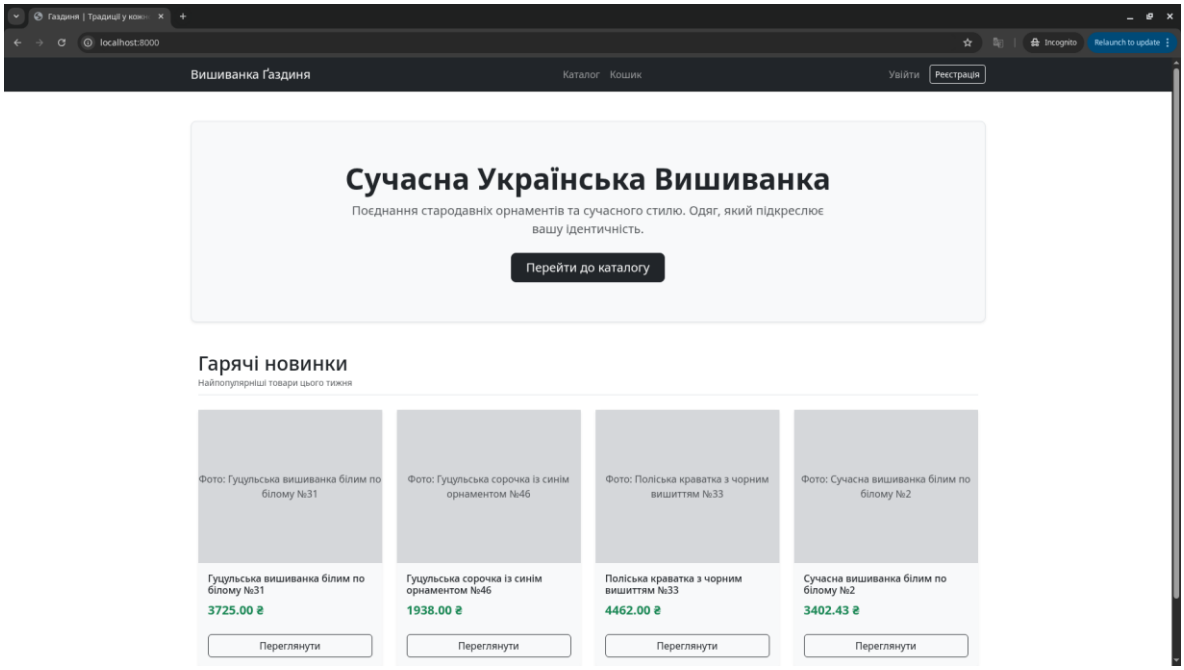


Рисунок 3.4 - Головна сторінка вебсайту та блока рекомендацій.

Джерело: розроблено автором

Основним елементом є блок «Рекомендовано для вас». Оскільки система працює в режимі SSR, під час генерації сторінки сервер автоматично перевіряє статус автентифікації користувача:

– якщо користувач авторизований, сервер зчитує з бази даних MySQL розраховані Celery-воркером персональні рекомендації (алгоритм k-NC) і віддає повністю згенеровану HTML-сторінку з картками цих товарів;

– якщо користувач є анонімним відвідувачем, у блок динамічно підставляються 4 найпопулярніші товари на основі загальної статистики логу interactions за останній місяць.

Перехід до каталогу здійснюється через верхнє навігаційне меню. На сторінці каталогу користувачу доступний повний асортимент традиційного одягу, розбитий за допомогою пагінатора по 12 карток на одну сторінку.

Для пошуку одягу користувач взаємодіє з бічною панеллю фільтрації, де доступні такі елементи керування:

– категоріальне меню: вибір конкретної категорії (наприклад, чоловічі вишиванки або сукні);

– ціновий діапазон: введення мінімальної та максимальної бажаної вартості у текстові поля;

– сортування: вибір критерію впорядкування (за новинками, за зростанням або спаданням ціни).

Після встановлення параметрів користувач натискає кнопку «Застосувати». Оскільки клієнтський JavaScript не використовується, ця дія відправляє стандартну HTML-форму методом GET на сервер. Сервер Django обробляє параметри в представленні ProductListView, виконує фільтрацію через ORM і повертає оновлену сторінку.

Завдяки впровадженому методу urlencode(), при переході користувача по номерах сторінок у нижній панелі навігації (пагінації), всі обрані критерії фільтрації зберігаються в URL-рядку, запобігаючи скиданню налаштувань пошуку.

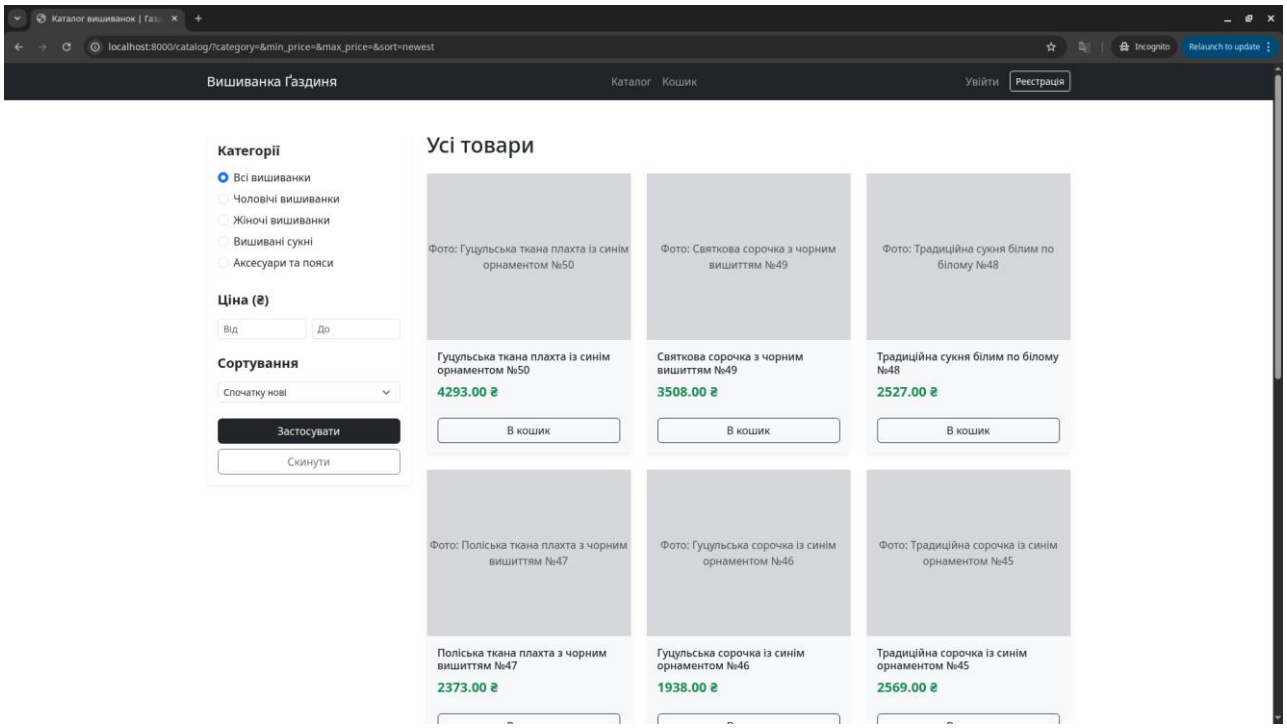


Рисунок 3.5 - Каталог товарів та панелі фільтрації.

Джерело: розроблено автором

Для створення персонального профілю користувач переходить за посиланням «Реєстрація». На екрані з'являється форма з полями введення імені користувача (ID) та електронної пошти. Після заповнення даних та натискання кнопки підтвердження, система виконує HTTP POST-запит.

У разі успішної валідації на стороні сервера, Django створює новий запис у таблиці users, автоматично авторизує користувача за допомогою сесійного механізму та перенаправляє його на головну сторінку, де блок рекомендацій одразу починає адаптуватися під новий профіль. Форма входу (Login) працює за аналогічним принципом, відновлюючи історію попередніх сесій клієнта.

На картці кожного товару в каталозі розміщено кнопку «В кошик». Натискання цієї кнопки ініціює відправку POST-запиту за допомогою форми безпосередньо на контролер cart_add. Програма додає ідентифікатор товару до

словника сесії на сервері та повертає користувачу сторінку каталогу з оновленим лічильником товарів у верхній панелі.

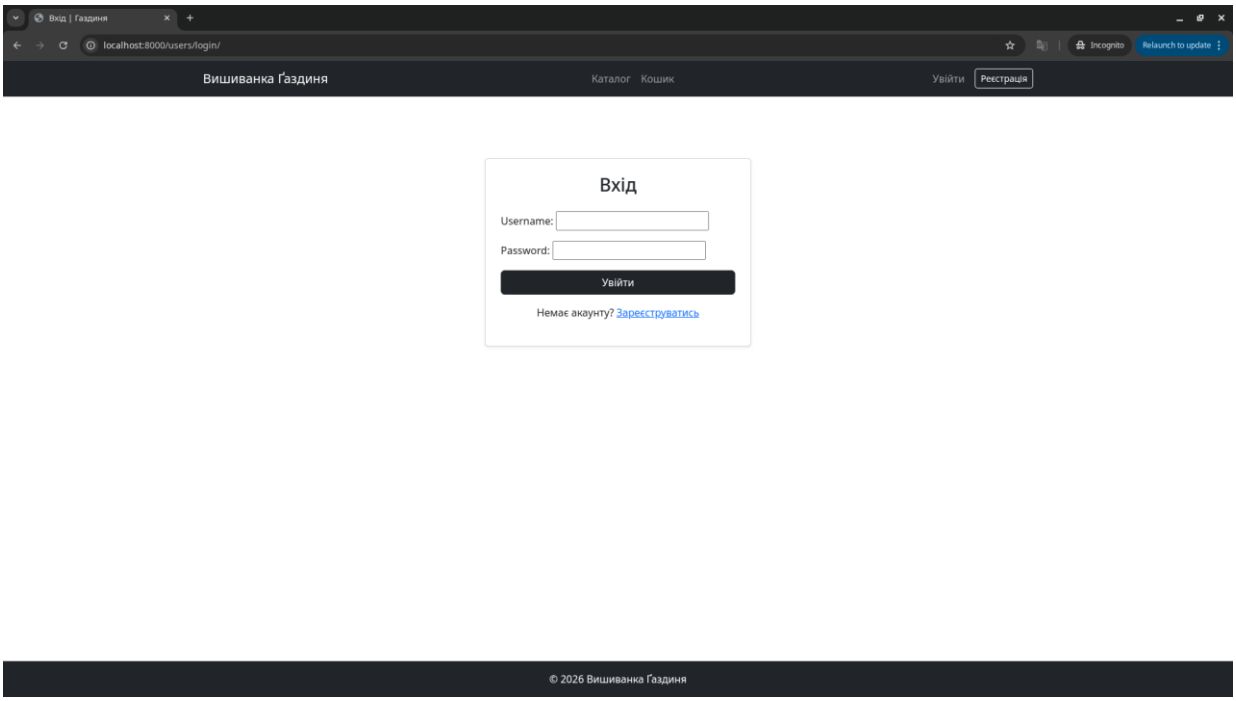


Рисунок 3.6 - Знімок екрану із відображенням форми автентифікації користувача.

Джерело: розроблено автором

При переході на сторінку кошика (Cart Detail) користувачу виводиться зведена таблиця, згенерована сервером на основі поточного стану сесії. Вона відображає:

– назву та зображення обраних моделей традиційного одягу;

– поточну ціну за одиницю (яка динамічно розраховується воркером Celery на основі коефіцієнта еластичності PED у фоновому режимі);

– кількість одиниць товару та підсумкову вартість кожної позиції;

– користувач може видалити товар, натиснувши на відповідне посилання (що викликає контролер cart_remove та перенаправляє назад на оновлений кошик), або повернутися до каталогу.

Також на сторінці розміщено функціональну кнопку «Оформити замовлення» для переходу до фінального етапу транзакції.

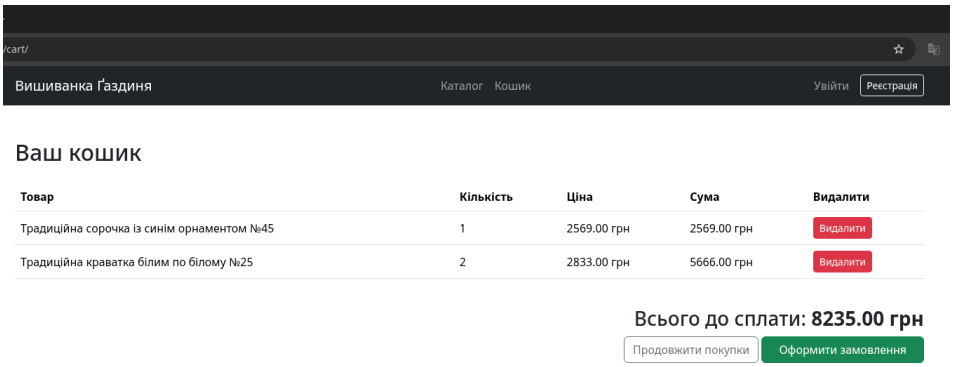


Рисунок 3.7 - Знімок екрану із відображенням інтерфейсу сесійного кошика покупця.

Джерело: розроблено автором

Висновки до розділу 3

На основі обраного технологічного стеку (Python 3.12, фреймворк Django 5.0, СУБД MySQL 8.0) успішно розгорнуто робоче програмне середовище MVP-прототипу в ізольованому контейнеризованому середовищі Docker. Завдяки декомпозиції системи на автономні застосунки (users, catalog, orders, analytics) та використанню інструментарію Django ORM забезпечено гнучку архітектуру, надійну обробку транзакцій та сесій покупця.

Спроєктовано та впроваджено асинхронний шар обробки даних на базі черги задач Celery, брокера повідомлень Redis та бібліотеки наукових обчислень NumPy. Тестування взаємодії сервісів підтвердило, що винесення ресурсоємних матричних розрахунків косинусної подібності (алгоритм k-НС) та цінової еластичності (PED) у фонові процеси надійно захищає основний вебсервер від перевантажень, гарантуючи високу швидкість відгуку користувацького інтерфейсу.

Проведено функціональне тестування прототипу методом «чорної скриньки» за критичними сценаріями, що підтвердило повну працездатність інтерфейсних рішень: багатокритеріальної фільтрації, пагінації, сесійного кошика та модулів авторизації. Доведено коректність обробки архітектурою виняткових ситуацій (крайових випадків), зокрема успішне вирішення

проблеми «холодного старту» для нових користувачів та запобігання критичним помилкам ділення на нуль (ZeroDivisionError) при розрахунку оптимізації статичних цін.

Розроблено детальну інструкцію користувача з ілюстраціями основних екранних форм, інтерфейс яких побудовано за принципом Server-Side Rendering (SSR). Описаний функціонал демонструє практичну цінність та готовність розробленої системи до використання. Також визначено інженерні перспективи подальшого розвитку продукту, такі як автоматизація запуску воркерів через планувальник та підключення транзакційних платіжних шлюзів.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи успішно розв’язано комплексне завдання щодо проєктування та розробки інтелектуального вебсайту інтернет-магазину традиційного одягу. Розроблений MVP-прототип програмного продукту підтверджує можливість динамічної адаптації до потреб клієнта та коливань ринкового попиту завдяки впровадженню інструментів глибокої персоналізації та автоматизованого ціноутворення.

В основу платформи покладено багатошарову архітектуру, розгорнуту в ізольованому контейнеризованому середовищі Docker із застосуванням фреймворку Django та реляційної СУБД MySQL. Ключовим інженерним досягненням стало проєктування слабкозв'язної системи, де транзакційна бізнес-логіка електронної комерції (навігація, сесійний кошик, оформлення замовлень) відокремлена від ресурсоємних математичних розрахунків. Імплементація асинхронного шару на базі брокера повідомлень Redis та черги задач Celery дозволила делегувати важкі обчислення у фонові воркери, гарантуючи високу швидкодію клієнтського інтерфейсу без блокування основного потоку запитів.

Інтелектуальне ядро програмного комплексу реалізоване на базі двох оптимізованих алгоритмів. Для системи персоналізованих рекомендацій застосовано метод колаборативної фільтрації k-найближчих сусідів, який знаходить приховані закономірності за допомогою розрахунку косинусної подібності користувацьких взаємодій. Для управління вартістю товарів розроблено модуль динамічного ціноутворення, що спирається на економіко-математичну модель цінової еластичності попиту (PED). Матричні обчислення цих моделей оптимізовано завдяки векторалізації за допомогою бібліотеки NumPy, а вразливості алгоритмів, такі як проблема «холодного старту» для нових відвідувачів та можливі помилки ділення на нуль, були успішно компенсовані архітектурними обробниками.

Працездатність розробленого MVP-прототипу підтверджена результатами функціонального й модульного тестування, які засвідчили стабільну взаємодію між вебсервером, базою даних та фоновими процесами, а також коректну роботу користувацьких модулів. Практичне значення одержаних результатів полягає у створенні масштабованої архітектурної бази, яка дозволить інтернет-магазину алгоритмічно автоматизувати управління ціновою політикою для максимізації прибутку та стимулювання збуту, водночас забезпечуючи покупцям релевантний персоналізований досвід.

Вихідний код розробленого програмного продукту та супровідна документація відкриті для рецензування і розміщені у публічному репозиторії на GitHub за посиланням: https://github.com/Rostislav-fargs/vyshyvanka-gazdynya .

ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАННЯ

1. Django: web framework. URL: https://www.djangoproject.com/ (дата звернення: 02.04.2026).
2. MySQL: relational database management system. URL: https://www.mysql.com/ (дата звернення: 02.04.2026).
3. Etnodim: інтернет-магазин сучасного етно-одягу. URL: https://etnodim.ua/ (дата звернення: 02.04.2026).
4. Vilni: концептуальний магазин традиційного одягу. URL: https://vilni.store/ (дата звернення: 02.04.2026).
5. Oracle. Програмне забезпечення як послуга (SaaS): що це таке. URL: https://www.oracle.com/ua/applications/what-is-saas/ (дата звернення: 02.04.2026).
6. Rozetka: найбільший універсальний маркетплейс України. URL: https://rozetka.com.ua/ (дата звернення: 02.04.2026).
7. Docker: платформа для контейнеризації. URL: https://www.docker.com/ (дата звернення: 30.04.2026).
8. dbdiagram.io: інструмент для створення ER-діаграм. URL: https://dbdiagram.io/ (дата звернення: 30.04.2026).
9. Leskovec J., Rajaraman A., Ullman J. D. Mining of Massive Datasets. 2nd ed. Cambridge: Cambridge University Press, 2014. 508 p. URL: http://www.mmds.org/ (дата звернення: 03.05.2026).
10. Taylor T., Greenlaw S. A., Shapiro D. Principles of Economics 2e. Houston: OpenStax, 2017. 1150 p. URL: https://openstax.org/details/books/principles-economics-3e (дата звернення: 03.05.2026).
11. Redis Documentation / Redis io. 2026. URL: https://redis.io/docs/ (дата звернення: 10.05.2026).
12. Celery 5.4.0 documentation / Celery Project. 2026. URL: https://docs.celeryq.dev/ (дата звернення: 10.05.2026).

13. NumPy: scientific computing package. URL: https://numpy.org/ (дата звернення: 15.05.2026).
14. PyCharm 2026.1.1: integrated development environment. URL: https://www.jetbrains.com/pycharm/ (дата звернення: 17.05.2025).
15. Wappalyzer: web technology profiler. URL: https://www.wappalyzer.com/ (дата звернення: 30.04.2026).