ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Веб платформа онлайн-магазину взуття з автоматизованим чат-ботом для підтримки користувачів»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки

освітньо-професійної програми Штучний інтелект

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Віталій ФЕДОРІВ

Виконав: здобувач вищої освіти гр. ШІД-42
Віталій ФЕДОРІВ

Керівник: Олександр РАБОТА
викладач,

Рецензент:
науковий ступінь,
вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту Ступінь

вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту

_____Ольга ЗІНЧЕНКО
«_____» _____2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Федорів Віталій Олегович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Веб-платформа онлайн-магазину взуття з автоматизованим чат-ботом для підтримки користувачів»

керівник кваліфікаційної роботи Олександр РАБОТА викладач,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025р. № 56

2. Строк подання кваліфікаційної роботи «23» травня 2025р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, специфікації та документація з HTML, CSS, JavaScript, JSON, Python та SQL, вимоги створення аналізатора користувача в чаті або інформаційному каналі.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Огляд технологій для розробки веб-застосунку з аналізом емоційного стану контексту користувача чату або інформаційного каналу використовуючи мову програмування PYTHON.

Вибір технологій для реалізації аналізатора текстів: HTML, CSS, JavaScript, JSON, Python та SQL.

Архітектура аналізатора текстів: .

Інтеграція з базою даних та аналізатором

Забезпечення безпеки та масштабованості системи

5. Перелік графічного матеріалу: *презентація*
   1. Схема архітектури системи
   2. Інтерфейс користувача онлайн-консультант
   3. Приклад коду клієнтської та серверної частини

6. Дата видачі завдання «25» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапу виконання кваліфікаційної роботи	Період реалізації	Примітка
1	Вивчення літературних джерел та стандартів, що стосуються веб-розробки та систем штучного інтелекту	25.02.2025 – 05.03.2025	Виконав
2	Ознайомлення з актуальними технологіями створення чат-ботів і засобів інтеграції з Django	06.03.2025 – 12.03.2025	Виконав
3	Вибір мов програмування, бібліотек, фреймворків і технологій для реалізації онлайн-магазину та чат-бота	13.03.2025 – 17.03.2025	Виконав
4	Розробка загальної структури та архітектури майбутньої системи	18.03.2025 – 22.03.2025	Виконав
5	Побудова та налаштування бази даних, моделей і взаємозв’язків у Django	23.03.2025 – 29.03.2025	Виконав
6	Розробка інтерфейсу користувача, основних сторінок магазину та реалізація логіки взаємодії з чат-ботом	30.03.2025 – 07.04.2025	Виконав
7	Підготовка графічних матеріалів, схем, скріншотів, оформлення презентації	29.04.2025 – 12.05.2025	Виконав
8	Остаточне оформлення та перевірка роботи перед подачею	13.05.2025 – 23.05.2025	Виконав

Здобувач вищої освіти _____ Віталій ФЕДОРІВ
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Олександр РАБОТА
(підпис) *(Ім’я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступення бакалавр: 55 стор., 2 табл., 31 рис., 26 джерел,.

Мета роботи: Розробити зручну, адаптивну та функціональну веб-платформу для електронної комерції, що дозволяє реалізувати процес продажу взуття із вбудованим інтелектуальним чат-ботом для консультації користувачів у реальному часі.

Об'єкт дослідження: Веб-системи електронної комерції.

Предмет дослідження: Методи інтеграції штучного інтелекту у функціонал веб-додатків для покращення взаємодії з користувачами.

Короткий зміст роботи: У процесі дослідження було проаналізовано основні принципи створення сучасних веб-платформ із використанням мов програмування Python та HTML, а також фреймворків Django і Bootstrap. Розглянуто варіанти реалізації сесійного кошика, фільтрації товарів, системи відгуків і підключення API чат-бота з використанням OpenAI. Реалізовано функціонал замовлення, додавання у вибране, перегляду історії покупок, а також система керування банерами. Проведено тестування ключових функцій, включаючи захист сесій, коректне обчислення суми замовлення та реагування чат-бота на запити користувача. Робота підтверджує доцільність впровадження автоматизованих консультантів на платформах електронної комерції.

Ключові слова: ВЕБ-ПЛАТФОРМА, ІНТЕРНЕТ-МАГАЗИН, ЧАТ-БОТ, DJANGO, PYTHON, BOOTSTRAP, ЕЛЕКТРОННА КОМЕРЦІЯ, OPENAI, ІНТЕРФЕЙС КОРИСТУВАЧА.

ABSTRACT

The qualification project is dedicated to the development of a web platform for an online shoe store, which includes an automated chatbot for user support. The explanatory note contains a total of 55 pages, includes 31 figures, 2 tables, and 26 references.

Objective of the work: To develop a user-friendly, adaptive, and functional web platform for e-commerce that enables the sale of footwear with an integrated intelligent chatbot for real-time user assistance.

Object of research: E-commerce web systems.

Subject of research: Methods of integrating artificial intelligence into the functionality of web applications to improve user interaction.

Summary:

The research analyzes the main principles of building modern web platforms using programming languages such as Python and HTML, as well as the Django and Bootstrap frameworks. Implementation options for a session-based shopping cart, product filtering, review system, and chatbot API integration using OpenAI are considered. The functionality includes order placement, adding to favorites, viewing purchase history, and banner management. Key features have been tested, including session protection, accurate order total calculation, and chatbot responsiveness to user queries. The work confirms the feasibility of implementing automated consultants on e-commerce platforms.

Keywords: WEB PLATFORM, ONLINE STORE, CHATBOT, DJANGO, PYTHON, BOOTSTRAP, E-COMMERCE, OPENAI, USER INTERFACE.

ЗМІСТ

Вступ

У світі електронні технології стали повсякденною частиною нашого життя і до того всього це чудово видно у сфері електронної комерції. Онлайн-магазини грають велику роль у забезпеченні доступу користувачів до широкого асортименту товарів, не дивлячись на їхнє місце розташування. Розвиток інтернет-продаж вимагає на постійній основі вдосконалення веб-платформ. Декілька із багатообіцяючих напрямків є інтеграція розумних систем підтримки користувачів, включаючи – чат-ботів.

Враховуючи високу конкуренцію на ринку, для компанії дуже важливо забезпечити комфорт користувача, можливість доступу до швидкої інформації про товар, якісне оформлення замовлення та зворотній зв’язок. Покупці прагнуть знайти моделі, які підходять за розміром, виглядом та функціональністю, не виходячи з дому.

Мета даної дипломної роботи закладається в тому, щоб розробити комфортний функціонал веб-платформи для продажу взуття, яка має підтримку автоматизованого чат-бота для допомоги користувачам. У рамках цієї роботи були проаналізовані інші інтернет-магазини, знайдені їх недоліки і впроваджені покращення вже у цей проект. Було розроблено зручний інтерфейс, була пророблена робота над функціональність сайту та впровадження чат-бота для взаємодії з клієнтами.

Вибір цієї теми зумовлений тим, що відбувається стрімкий розвиток IT-технологій, з’являється попит на автоматизовані сервіси підтримки покупців та необхідністю впровадження інтелектуальних рішень у традиційні комерційні моделі.

Структура цієї дипломної роботи включає в себе вступ, чотири розділи, висновки до кожного розділу, загальний висновок до дипломної роботи, список використаних джерел та додатки. У першому розділі описуються теоретичні основи, як будується веб-платформа і чат-бота. У другому розділі показується аналіз вимог, опис інструментів, які були використанні. У третьому розділі

показано тестування функціоналу платформи та проаналізовано її ефективність.

1 АНАЛІЗ СТАНУ РОЗВИТКУ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ВЕБ-ПЛАТФОРМИ

1.1 Стан розвитку IT в електронній комерції

У нашому світі цифрові технології стають основою спілкування між бізнесом та користувачами.

У сучасному інформаційному просторі, технології грають дуже важливу роль у створенні унікальних способів ведення бізнесу. На цей день конкуренція між інтернет-магазинами сильно зростає, оскільки багато хто починає впроваджувати щось універсальне, шукають нові рішення для комфорту користувачів. Деякі магазини мають вагому перевагу над іншими в утворенні автоматизованого і модернізованого чат-боту. При такому рішенні клієнту не приходиться очікувати тривалу по часу відповідь від оператора і він зразу отримує відповіді на свої питання за рахунок чат-боту. До того ж всього це підвищує ефективність роботи веб-платформи загалом. Інтернет-магазин з інтегрованим чат-ботом дозволяє краще розуміти клієнта, його потреби та моментально на них реагувати.

За останні роки електронна комерція зросла в розвитку. Особливо це стосується хмарних технологій, які дозволяють вирішувати масштабні задачі без необхідності розгортання власної інфраструктури. Тим більше в наш час з використанням мобільного інтернету, люди можуть оформлювати покупки в інтернет-магазинах будучи будь-де і будь-коли. Разом з цим всім, клієнти очікують чогось нового, щось те, що буде більш зручнішим, швидшим.

Якщо ми говоримо саме про Україну, то інтернет-торгівля розвивається тут все більше і більше, вона активно розвивається. Можна все частіше бачити, як люди поєднують використання веб-платформ та походження до звичайного магазину.

1.2 Аналіз існуючих веб-магазинів

Так званні «розумні» системи підтримки користувачів стали над важливою частиною для інтернет-магазинів. Вони значно покращують роботу веб-

платформ, знижують навантаження на службу підтримки. Чат-боти демонструють здатність ефективно оброблять сотні запитів одночасно, з чим до речі людям дуже важко справитись. Надають допомогу 24/7, що теж є великим плюсом.

Технології постійно розвиваються, створювати веб-платформи можна вже на декількох IT-мовах. Якщо ми будемо говорити про HTML, CSS, JavaScript, то ці мови більше використовують для створення динамічних та зрозумілих інтерфейсів. У той самий час, якщо нам потрібно розробити серверну частину веб-платформи, то вже будемо використовувати для зручності мову Python і до неї фреймворк Django.

Станом на сьогодні існує вже тисячі інтернет-магазинів, кожен з яких по своєму особливий. Можна назвати навіть деякі з ним, по типу Амазон, АліЕкспресс, вони показують наскільки веб-магазини можуть бути масштабними по своєму функціоналу, по кількості своїх товарів, по своїй складній системі.

На ринку України теж є масштабні проекти, такі як: Інтертоп, Розетка (рис. 1.2)

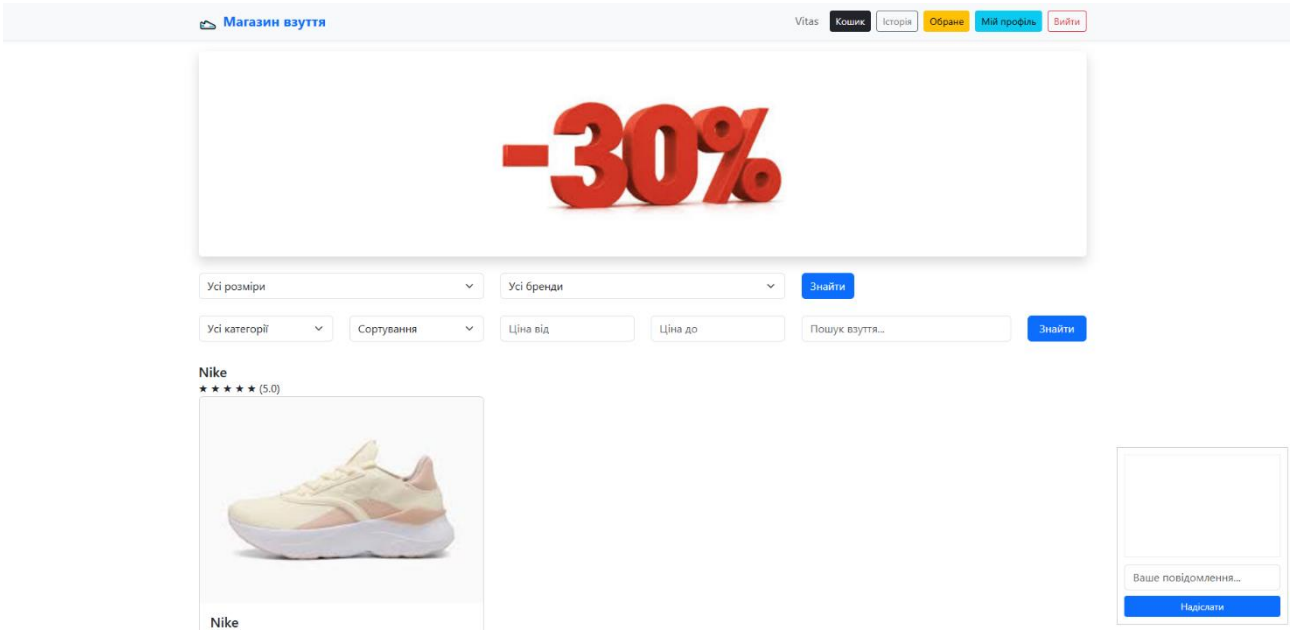


Рисунок 1.2 - Головна сторінка онлайн магазину-взуття

1.3Обґрунтування вибору інструментальних засобів

Онлайн-магазин містить такі основні функціональні модулі, як каталог товарів, система фільтрації, модулі замовлень, кошик, система обліку користувачів (рис. 1.3.1)(рис. 1.3.2)

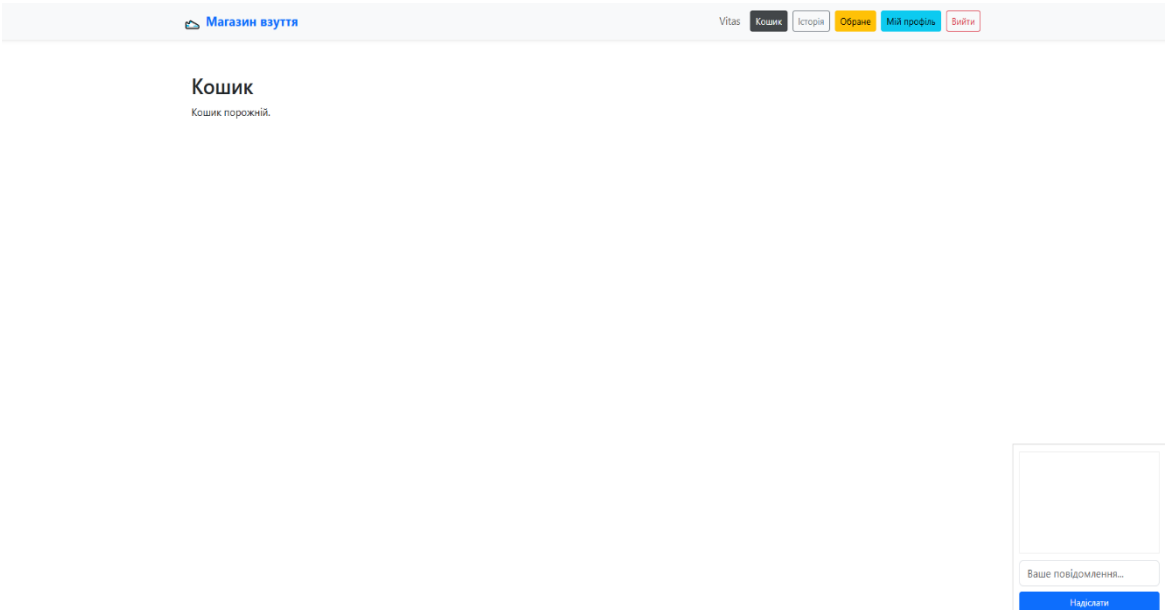


Рисунок 1.3.1 - Сторінка кошику

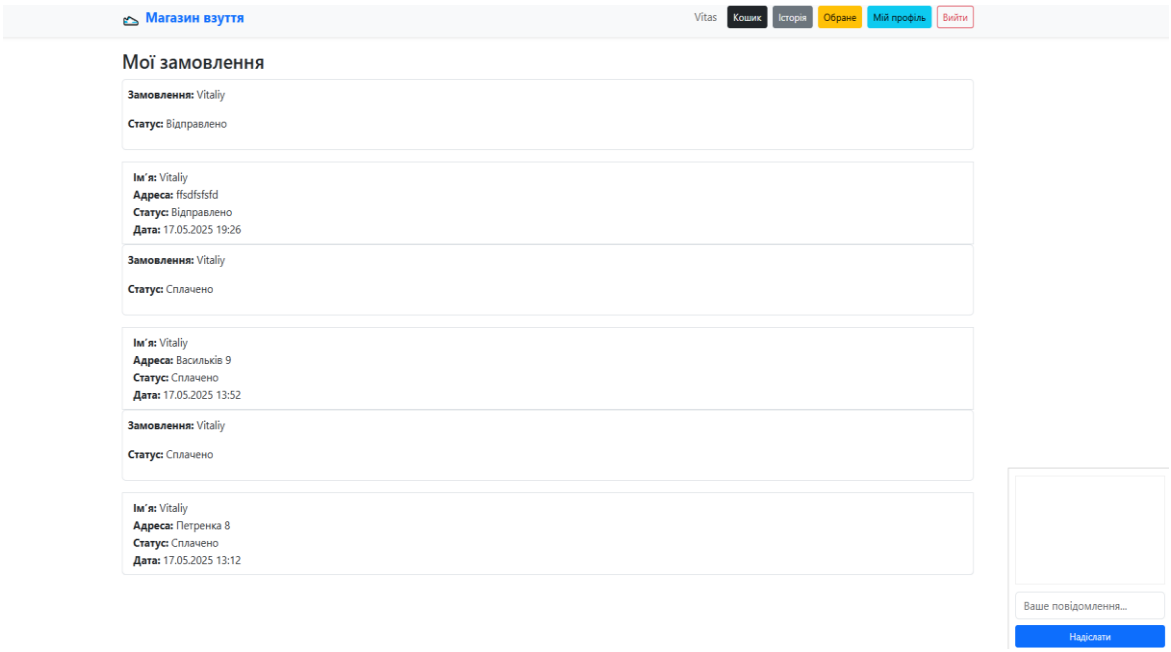


Рисунок 1.3.2 - Сторінка історії замовлень

Під час розробки інтернет-магазину використовувався фреймворк Django, який використовується при підтримці мови Python. З основ це те, що Django забезпечує високий рівень безпеки, має вбудовану ORM-систему для роботи з

базами даних, а також зручну систему шаблонів, яка дозволяє швидко створювати динамічні HTML-сторінки.

Також мною обирався Bootstrap для фронтенду, який дозволяє адаптувати інтерфейс під різні пристрої без потреби розробляти окремі стилі для кожного з них. Використання jQuery або JavaScript дозволяє реалізувати такі елементи, як фільтри, слайдери, та динамічні списки.

Для реалізації чат-бота доцільним є інтеграція з API OpenAI або використання бібліотек Python для NLP. Це дає змогу реалізувати підтримку клієнтів, здатну реагувати на запити, допомагати у виборі товару та супроводжувати користувача на всіх етапах замовлення.

1.4Постановка завдання

Взагалі особлива увага приділяється тому, що платформа зобов’язана забезпечити масштабність, надійність і безпеку для користувача, а також зберігання конфіденційних даних цього користувача і його історії замовлень.

Основними завданнями цієї роботи являються:

- Проектування архітектури веб-платформи;
- Розробка бази даних для збереження інформації про товари, замовлення, користувачів;
- Реалізація фронтенду із застосуванням адаптивного дизайну;
- Тестування системи та оцінка її ефективності
- Забезпечення базового рівня безпеки
- Інтеграція чат-бота
- Розробка функціональних модулів

Якщо виконати усі ці задачі, то це дозволить створити дійсно сучасну і зручну для використання платформу.

1.5 Висновок

У цьому розділі було розглянуто, в якому стані наразі знаходяться веб-платформи, проаналізовано їх та розглянули інструменти за рахунок, яких і було створено ці інтернет-магазини. Також обговорили вибір інструментів для цього проекту та сформулювали задачі, які необхідно вирішити в ході розробки цього

інтернет-магазину. Отриманні результати стали основою для формування технічного завдання та реалізації подальших етапів роботи.

2 ПРОЄКТУВАННЯ ВЕБ-ПЛАТФОРМИ ОНЛАЙН-МАГАЗИНУ ВЗУТТЯ

2.1 Архітектура платформи

Щоб успішно розробити хорошу веб-платформу інтернет-магазину взуття, потребується ретельний підхід до проектування її архітектури. Архітектура веб-платформи була побудована за принципи модульності, безпеки, масштабованості.

Основні компоненти архітектури:

- Front-end: Була розроблена за допомогою HTML, CSS, JavaScript і використовувався Bootstrap для адаптивної верстки.
- База даних: Було обрано PostgreSQL, що дозволяє зберегти інформацію щодо товарів, бренди, категорії, замовлення.
- Back-end: Використовувалась мова Python з допомогою фреймворку Django, який забезпечує якісну розробку, захист від всяких вразливостей.

Ще також відповідає за роботу кошика, замовлення, авторизація користувача, фільтрацію товарів (рис. 2.1.1)

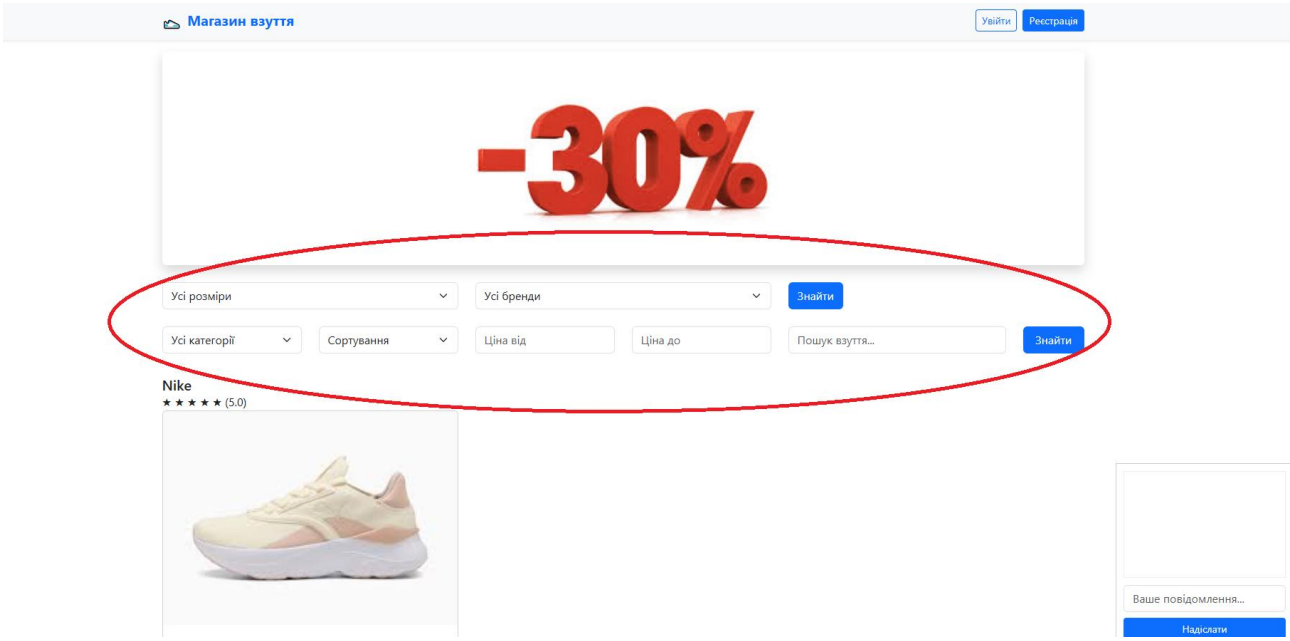


Рисунок 2.1.1 - Фільтрація товарів

Адмін-панель: Django автоматично створює інтерфейс адміністратора для управління вмістом. Адміністратор може додати товар, змінювати його ціну,

керувати знижками, додавати користувачів, додавати банери, переглядати замовлення. Зі всім цим він теж може редагувати і видаляти (рис. 2.1.2)

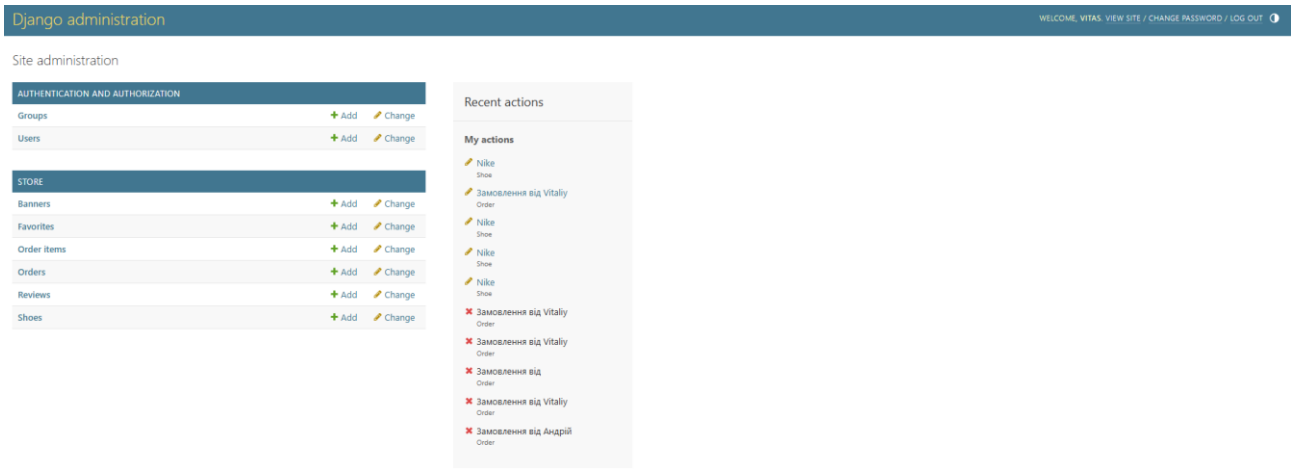


Рисунок 2.1.2 - Сторінка адміністратора

- Чат-бот: Чат-бот створений як незалежний модуль, який обмінюється даними з сервером через API. Він приймає запити від користувача, обробляє їх і надсилає відповідь, сформовану на основі попередньо заданих сценаріїв або алгоритмів штучного інтелекту.

- Структура файлів проекту: store/ використовується для основи магазину, cart/ для кошика, chatbot/ для обробки запитів чат-бота, media/ для зображень товару, shoe_shop/ являється основною папкою, де зберігаються всі наступні

Структура системи зроблена таким чином, що враховуються можливості масштабування. Якщо навантаження на сервер будуть великими, запити можна рівно розподіляти між кількома копіями застосунку за допомогою таких інструментів, як Nginx або Gunicorn. Для покращення стабільності основні сервіси доцільно винести в Docker-контейнери, що також спрощує їхнє розгортання та обслуговування.

Щоб не втратити важливі дані, в платформі налаштоване щоденне збереження копій бази даних. Якщо щось раптом зламається або станеться збій — вся інформація залишиться в безпеці.

Також система фіксує всі дії користувачів. Це допомагає швидко знаходити помилки. Для цього використовуються стандартні інструменти Django або сторонні сервіси, як Sentry чи Loguru.

Усі частини платформи чітко розділені — кожна відповідає лише за свою задачу. Завдяки цьому сайт легко оновлювати, підтримувати та змінювати, якщо з'являться нові потреби.

Коли система поділена на частини, з нею легше працювати. Можна без проблем додавати нові функції — наприклад, показувати схожі товари або підключати інші сайти для продажу.

Сайт магазину взуття зроблений так, щоб працювати стабільно, бути безпечним і легко доповнюватись новими функціями в майбутньому.

Одна з важливих частин — це система, яка записує всі дії на сайті. Вона фіксує, що роблять користувачі, як оформлюються замовлення, і допомагає знайти та виправити помилки. У Django це налаштовується дуже просто — всі записи зберігаються у файлах.

Щоб сторінки відкривались швидко, частина з них (наприклад, з популярними товарами) зберігається в пам'яті на певний час. Це зменшує навантаження на сервер. Для цього використовується Redis або вбудовані засоби Django.

Ще одна зручна річ — це шаблони. Вони дозволяють відділити програмний код від зовнішнього вигляду сайту. Завдяки цьому оновлювати дизайн легко і швидко, без зайвих клопотів.

Сайт зроблений так, що його легко можна розширити. Якщо людей стане більше — його можна перенести на хмарні сервіси, наприклад AWS чи DigitalOcean, і розподілити навантаження між кількома серверами, щоб усе працювало швидко.

Щоб нічого не загубилось, база даних регулярно зберігається. Це налаштовується автоматично і дуже корисно, бо зберігає всю важливу інформацію: замовлення, облікові записи, відгуки.

Уся система продумана так, щоб не тільки виконувати свої функції, а ще й мати можливість рости, змінюватись і не ламатися при оновленнях чи збільшенні навантаження.

2.2 Проектування бази даних

Щоб сайт магазину працював добре, дуже важливо правильно налаштувати базу даних. Саме там зберігається вся інформація — про товари, замовлення, користувачів, відгуки, банери, кошик і багато іншого. У цьому розділі розглянемо, як побудована база даних, які таблиці вона містить, як вони пов’язані між собою, і як усе це працює в проекті через Django ORM.

Основні моделі бази даних

У Django кожен тип даних утворює окремий клас на Python, який автоматично перетворюється на таблицю в базі даних. Наприклад, є моделі для товарів (Shoe), замовлень (Order), товарів у замовленні (OrderItem), відгуків (Review), обраного (Favorite) і банерів (Banner). Усі ці моделі пов’язані між собою через зовнішні ключі, тому база працює як єдине ціле.

```python
class Shoe(models.Model):
    CATEGORY_CHOICES = [
        ('man', 'Чоловіче'),
        ('woman', 'Жіноче'),
        ('kid', 'Дитяче'),
    ]
    SIZE_CHOICES = [
        ('36', '36'), ('37', '37'), ('38', '38'), ('39', '39'),
        ('40', '40'), ('41', '41'), ('42', '42'), ('43', '43'), ('44', '44')
    ]
    BRAND_CHOICES = [
        ('Nike', 'Nike'), ('Adidas', 'Adidas'), ('Puma', 'Puma'),
        ('Reebok', 'Reebok'), ('New Balance', 'New Balance')
    ]

    size = models.CharField(max_length=3, choices=SIZE_CHOICES)
    brand = models.CharField(max_length=20, choices=BRAND_CHOICES)
    name = models.CharField(max_length=100)
    description = models.TextField()
    price = models.DecimalField(max_digits=7, decimal_places=2)
    image = models.ImageField(upload_to='shoes/')
    category = models.CharField(max_length=10, choices=CATEGORY_CHOICES, default='man')
    discount_percent = models.PositiveIntegerField(default=0, help_text="Знижка у відсотках")

    def __str__(self):
        return self.name

    def average_rating(self):
        reviews = self.reviews.all()
        if reviews.exists():
            return round(sum([r.rating for r in reviews]) / reviews.count(), 1)
        return 0
```

Рисунок 2.2.1 - Код написання моделі Shoe

Shoe — це модель, яка зберігає все, що треба знати про взуття: який розмір, бренд, опис, скільки коштує, яке фото, чи є знижка і до якої категорії належить. Вона також пов’язана з відгуками, тож можна побачити, наскільки товар подобається людям. У деяких полях є готові варіанти для вибору — це

зручно, коли додаєш товар в адмінці. А ще є метод, який одразу показує середню оцінку взуття (рис. 2.2.1)

```python
class Order(models.Model):
    STATUS_CHOICES = [
        ('processed', 'Оброблено'),
        ('shipped', 'Відправлено'),
        ('delivered', 'Доставлено'),
        ('paid', 'Сплачено'),
    ]

    user = models.ForeignKey(User, on_delete=models.CASCADE, null=True, blank=True)
    name = models.CharField(max_length=100)
    address = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    status = models.CharField(max_length=20, choices=STATUS_CHOICES, default='processed')

    def __str__(self):
        return f"Замовлення від {self.name}"
```

Рисунок 2.2.2 - Код написання моделі Order

Order зберігає інформацію про покупку: користувача, дату, статус (опрацьовано, сплачено, відправлено) (рис. 2.2.2)

```python
class OrderItem(models.Model)  (class) Shoe
    shoe = models.ForeignKey(Shoe, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=1)
    order = models.ForeignKey(Order, on_delete=models.CASCADE)

    def total_price(self):
        return self.shoe.price * self.quantity
```

Рисунок 2.2.3 - Код написання моделі OrderItem

OrderItem — це просто частина замовлення, де вказано, яке взуття вибрали і скільки пар. Вона прив’язана до замовлення, і якщо замовлення видалити — ця частина теж зникне автоматично. Так зручніше, і в базі не буде лишнього (рис. 2.2.3)

```python
class Favorite(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    shoe = models.ForeignKey('Shoe', on_delete=models.CASCADE)  # ← в лапках — УВАГА

    class Meta:
        unique_together = ('user', 'shoe')

    def __str__(self):
        return f"{self.user.username} ♥ {self.shoe.name}"
```

Рисунок 2.2.4 - Код написання моделі Favorite

Favorite — це модель, яка зберігає, яке взуття користувач додав у обране. Вона пов’язана і з користувачем, і з конкретною парою взуття. Завдяки налаштуванням запис про один і той самий товар не може з’явитися в обраному два рази — усе чітко і без дублю (рис. 2.2.4)

```python
class Review(models.Model):
    shoe = models.ForeignKey(Shoe, on_delete=models.CASCADE, related_name='reviews')
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    text = models.TextField()
    rating = models.IntegerField(choices=[(i, str(i)) for i in range(1, 6)])
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.user.username} - {self.rating}★"
```

Рисунок 2.2.5 - Код написання моделі Review

Review — це модель для відгуків. У ній зберігається, яку оцінку поставив користувач, що написав у коментарі та коли саме. Кожен відгук прив'язаний до конкретного товару і до людини, яка його залишила. Оцінка може бути від 1 до 5 — не більше і не менше (рис. 2.2.5)

```python
class Banner(models.Model):
    image = models.ImageField(upload_to='banners/')
    alt_text = models.CharField(max_length=100, verbose_name='Опис картинки')
    is_active = models.BooleanField(default=True, verbose_name='Активний')
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.alt_text
```

Рисунок 2.2.6 - Код написання моделі Banner

Banner — це модель для рекламних банерів. У ній зберігається картинка, текст і чи активний банер. Через адмінку можна легко додавати або вимикати банери. Це зручно для реклами й допомагає просувати товари на сайті (рис. 2.2.6)

Структура зв'язків

Усі моделі в базі пов'язані між собою через спеціальні поля. Це потрібно, щоб усе було пов'язано правильно і без повторів. Наприклад, якщо видалити користувача — автоматично видаляться його замовлення та відгуки. Це зручно і все працює чітко.

У результаті база зроблена так, щоб не було плутанини. Її легко розширювати, додавати нові штуки і просто керувати всім через адмінку.

2.3 Розробка структури користувацького інтерфейсу

Інтерфейс — це те, що бачить користувач на сайті. Він має бути зручним, простим і приємним на вигляд. Щоб зробити все красиво й щоб працювало на телефоні й комп'ютері, використовувався HTML, CSS, JavaScript і Bootstrap. Це сучасні мови, які допомагають швидко зробити сайт з нормальним дизайном.

Основне завдання інтерфейсу — зробити так, щоб користувачу було легко

знайти потрібне взуття. Щоб він міг переглядати каталог, шукати, фільтрувати товари, дивитися деталі, додавати в кошик і, якщо потрібно, написати чат-боту. Для цього всі ці частини винесені на окремі сторінки, які створюються за допомогою шаблонів Django.

Перше, що бачить людина, коли заходить у магазин. Тут показуються банери з акціями, список взуття, фільтри та сортування. Можна вибрати, що саме шукати — за категорією, брендом або ціною. Усе це зроблено просто, через звичайні форми. Якщо товарів багато, вони поділені на сторінки. А ще біля кожного товару видно оцінку у вигляді зірочок (рис. 2.3.1)(рис. 2.3.2)(рис. 2.3.3)(рис. 2.3.4)(рис. 2.3.5)

```
{% extends 'store/base.html' %}
{% load widget_tweaks %}
{% load static %}
{% load price_tags %}
{% block content %}

{% if banners %}
<div id="bannerCarousel" class="carousel slide mb-4" data-bs-ride="carousel">
  <div class="carousel-inner">
    {% for banner in banners %}
      <div class="carousel-item {% if forloop.first %}active{% endif %}">
        <img src="{{ banner.image.url }}" class="d-block w-100" alt="{{ banner.alt_text }}">
      </div>
    {% endfor %}
  </div>
  <button class="carousel-control-prev" type="button" data-bs-target="#bannerCarousel" data-bs-slide="prev">
    <span class="carousel-control-prev-icon"></span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-target="#bannerCarousel" data-bs-slide="next">
    <span class="carousel-control-next-icon"></span>
  </button>
</div>
{% endif %}


<!-- Слайдер банерів -->
<div id="promoCarousel" class="carousel slide mb-4" data-bs-ride="carousel">
  <div class="carousel-inner rounded shadow">
    <div class="carousel-item active">
      <img src="{% static 'store/banner1.jpg' %}" class="d-block w-100"
           style="max-height: 300px; object-fit: contain; background: ■#fff;" alt="banner1">
    </div>
    <div class="carousel-item">
      <img src="{% static 'store/banner2.jpg' %}" class="d-block w-100"
           style="max-height: 300px; object-fit: contain; background: ■#fff;" alt="banner2">
    </div>
  </div>
  <button class="carousel-control-prev" data-bs-target="#promoCarousel" data-bs-slide="prev">
    <span class="carousel-control-prev-icon"></span>
  </button>
  <button class="carousel-control-next" data-bs-target="#promoCarousel" data-bs-slide="next">
    <span class="carousel-control-next-icon"></span>
  </button>
</div>
```

Рисунок 2.3.1 - Код home.html банери

```
<!-- Фільтри -->
<form method="get" class="row mb-4">
  <div class="col">
    {{ form.size }}
  </div>
  <div class="col">
    {{ form.brand }}
  </div>
  <div class="col">
    <button type="submit" class="btn btn-primary">Знайти</button>
  </div>
</form>
<form method="get" class="row mb-4">
  <div class="col-md-2 mb-2">
    <select name="category" class="form-select">
      <option value="">Усі категорії</option>
      <option value="man">Чоловіче</option>
      <option value="woman">Жіноче</option>
      <option value="kid">Дитяче</option>
    </select>
  </div>
  <div class="col-md-2 mb-2">
    <select name="sort" class="form-select">
      <option value="">Сортування</option>
      <option value="price_asc">Ціна ↑</option>
      <option value="price_desc">Ціна ↓</option>
      <option value="name_asc">Назва А-Я</option>
      <option value="name_desc">Назва Я-А</option>
    </select>
  </div>
  <div class="col-md-2 mb-2">
    <input type="text" name="min_price" class="form-control" placeholder="Ціна від">
  </div>
  <div class="col-md-2 mb-2">
    <input type="text" name="max_price" class="form-control" placeholder="Ціна до">
  </div>
  <div class="col-md-3 mb-2">
    <input type="text" name="search" class="form-control" placeholder="Пошук взуття...">
  </div>
  <div class="col-md-1 mb-2">
    <button type="submit" class="btn btn-primary w-100">Знайти</button>
  </div>
</form>
```

Рисунок 2.3.2 - Код home.html Фільтри

```
<!-- Список взуття -->
<div class="row">
  {% for shoe in shoes %}
    <h5 class="card-title">{{ shoe.name }}</h5>

<!-- Середній рейтинг -->
<div>
    {% with rating=shoe.average_rating %}
        <span>
            {% for i in "12345" %}
                {% if i|add:'0' <= rating %}
                    ★
                {% elif i|add:'0' > rating and i|add:'0' <= rating|floatformat:0 %}
                    ★
                {% else %}
                    ☆
                {% endif %}
            {% endfor %}
            ({{ rating }})
        </span>
    {% endwith %}
</div>
    <div class="col-md-4 mb-4">
      <div class="card h-100">
        <img src="{{ shoe.image.url }}" class="card-img-top" alt="{{ shoe.name }}">
        <div class="card-body">
          <h5 class="card-title">{{ shoe.name }}</h5>

          {% if shoe.discount_percent > 0 %}
            <p>
              <del class="text-muted">{{ shoe.price }} грн</del><br>
              <span class="text-success fw-bold">
                {{ shoe.price|discount:shoe.discount_percent }} грн
              </span>
              <span class="badge bg-danger ms-2">-{{ shoe.discount_percent }}%</span>
            </p>
          {% else %}
            <p class="card-text">{{ shoe.price }} грн</p>
          {% endif %}
```

Рисунок 2.3.3 - Код home.html список взуття і середній рейтинг

```
        <!-- Обране -->
        {% if user.is_authenticated %}
          {% if shoe.is_favorite %}
            <a href="{% url 'remove_from_favorites' shoe.id %}" class="btn btn-sm btn-danger mb-2">♥ В обраному</a>
          {% else %}
            <a href="{% url 'add_to_favorites' shoe.id %}" class="btn btn-sm btn-outline-danger mb-2">♡ В обране</a>
          {% endif %}
        {% endif %}

        <a href="{% url 'add_to_cart' shoe.id %}" class="btn btn-sm btn-success w-100 mb-2">🛒 До кошика</a>

        <a href="{% url 'detail' shoe.id %}" class="btn btn-primary w-100">Детальніше</a>
      </div>
    </div>
  </div>
 {% empty %}
   <p>Нічого не знайдено.</p>
 {% endfor %}
</div>
<!-- Пагінація -->
{% if shoes.has_other_pages %}
<div class="d-flex justify-content-center mt-4">
  <nav>
    <ul class="pagination">
      {% if shoes.has_previous %}
        <li class="page-item">
          <a class="page-link" href="?page={{ shoes.previous_page_number }}">&laquo;</a>
        </li>
      {% endif %}

      {% for num in shoes.paginator.page_range %}
        {% if shoes.number == num %}
          <li class="page-item active"><span class="page-link">{{ num }}</span></li>
        {% else %}
          <li class="page-item"><a class="page-link" href="?page={{ num }}">{{ num }}</a></li>
        {% endif %}
      {% endfor %}
```

Рисунок 2.3.4 - Код home.html обране та пагінація

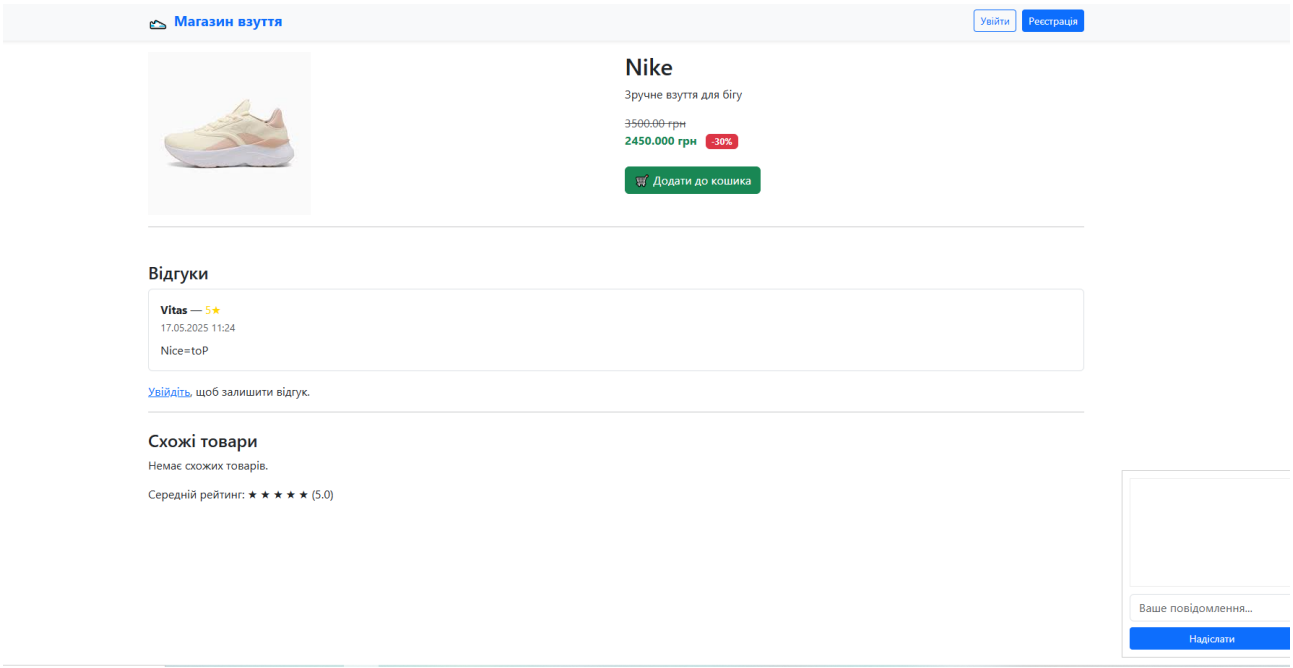


Рисунок 2.3.5 - Сторінка «Детальніше» товару

Кожне взуття на сайті виглядає як окрема картка: є фото, назва, ціна (зі знижкою або звичайна), і кнопки — «Детальніше», «Додати до кошика» та «Обране». Якщо ти зайшов у свій акаунт, то можеш додати товар в улюблене. Все це працює через спеціальні посилання та перевірку, чи ти ввійшов на сайт (рис. 2.3.6)(рис. 2.3.7)(рис. 2.3.8)

```
{% extends 'store/base.html' %}
{% load static %}
{% load price_tags %}
{% block content %}

<div class="row">
  <div class="col-md-6">
    <img src="{{ shoe.image.url }}" class="img-fluid" alt="{{ shoe.name }}">
  </div>
  <div class="col-md-6">
    <h2>{{ shoe.name }}</h2>
    <p>{{ shoe.description }}</p>

    {% if shoe.discount_percent > 0 %}
      <p>
        <del class="text-muted">{{ shoe.price }} грн</del><br>
        <span class="text-success fw-bold">
          {{ shoe.price|discount:shoe.discount_percent }} грн
        </span>
        <span class="badge bg-danger ms-2">-{{ shoe.discount_percent }}%</span>
      </p>
    {% else %}
      <h4 class="text-success fw-bold">{{ shoe.price }} грн</h4>
    {% endif %}

    <form method="post">
      {% csrf_token %}
      <input type="hidden" name="add_to_cart" value="1">
      <button type="submit" class="btn btn-success mt-2">🛒 Додати до кошика</button>
    </form>
  </div>
</div>

<hr>
```

Рисунок 2.3.6 Код detail.html знижки, кнопка додавання до кошика

```
<h4 class="mt-5">Відгуки</h4>
{% for review in reviews %}
  <div class="border rounded p-3 mb-3">
    <div class="d-flex justify-content-between">
      <div>
        <strong>{{ review.user.username }}</strong> —
        <span style="color: gold;">{{ review.rating }}★</span><br>
        <small class="text-muted">{{ review.created_at|date:"d.m.Y H:i" }}</small>
      </div>
      {% if review.user == user %}
        <div>
          <a href="{% url 'edit_review' review.id %}" class="btn btn-sm btn-outline-primary me-1">✏️</a>
          <a href="{% url 'delete_review' review.id %}" class="btn btn-sm btn-outline-danger"
             onclick="return confirm('Видалити відгук?');">🗑</a>
        </div>
      {% endif %}
    </div>
    <p class="mt-2 mb-0">{{ review.text }}</p>
  </div>
{% empty %}
  <p class="text-muted">Ще немає відгуків. Будьте першим!</p>
{% endfor %}

{% if user.is_authenticated %}
  <hr>
  <h5>Залишити відгук</h5>
  <form method="post" class="mt-3">
    {% csrf_token %}
    <input type="hidden" name="add_review" value="1">
    <div class="mb-2">
      <label>Оцінка:</label>
      <select name="rating" class="form-select w-auto d-inline">
        {% for i in rating_choices %}
          <option value="{{ i }}">{{ i }}★</option>
        {% endfor %}
      </select>
    </div>
    <div class="mb-3">
      <textarea name="text" class="form-control" placeholder="Ваш відгук..." required></textarea>
    </div>
    <button type="submit" class="btn btn-primary">Надіслати</button>
  </form>
{% else %}
  <p class="mt-3">
    <a href="{% url 'login' %}">Увійдіть</a>, щоб залишити відгук.
  </p>
{% endif %}
<hr>
```

Рисунок 2.3.7 - Код detail.html Відгуки

```
<h4 class="mt-4">Схожі товари</h4>
<div class="row">
    {% for item in similar_shoes %}
        <div class="col-md-3">
            <div class="card mb-3">
                <img src="{{ item.image.url }}" class="card-img-top" alt="{{ item.name }}">
                <div class="card-body">
                    <h5 class="card-title">{{ item.name }}</h5>
                    <p class="card-text">{{ item.price }} грн</p>
                    <a href="{% url 'shoe_detail' item.id %}" class="btn btn-sm btn-primary">Детальніше</a>
                </div>
            </div>
        </div>
    {% empty %}
        <p>Немає схожих товарів.</p>
    {% endfor %}
</div>
<p>
    Середній рейтинг:
    {% with rating=shoe.average_rating %}
        {% for i in "12345" %}
            {% if i|add:'0' <= rating %}
                ★
            {% else %}
                ☆
            {% endif %}
        {% endfor %}
        ({{ rating }})
    {% endwith %}
</p>


{% endblock %}
```

Рисунок 2.3.8 - Код detail.html Схожі товари

Сторінка з товаром показує всю потрібну інформацію про взуття — опис, ціну, чи є знижка. Тут можна додати товар у кошик, подивитись відгуки інших людей або написати свій. Якщо є оцінки, то видно зірочки. Усе це працює завдяки коду, який перевіряє, що саме показати і що робити, коли ти натискаєш кнопку.

Блок відгуків працює динамічно. Якщо людина увійшла в акаунт — вона бачить форму, де можна вибрати оцінку (від 1 до 5) і написати свій коментар. Кожен відгук показує, хто його написав, яку поставив оцінку, коли саме, і якщо це твій відгук — ти можеш його змінити або видалити.

На сторінці можна побачити схожі товари — це зручно, якщо хочеш подивитися інші варіанти. А ще є зручне листання сторінок, якщо товарів багато. Сайт не гальмує, усе працює швидко. Це зроблено за допомогою простих інструментів Django і Bootstrap.

Сайт зручно працює і на комп’ютері, і на телефоні. Завдяки Bootstrap усе красиво підлаштовується під будь-який розмір екрана — нічого не з’їжджає і не ламається.

Загалом інтерфейс зроблений так, щоб людині було легко користуватися сайтом. Усе виглядає сучасно, просто і зрозуміло — це важливо, щоб людям хотілося повертатись і щось купувати.

2.4 Реалізація функціональних модулів

Функціональні модулі — це основні частини сайту, з якими взаємодіє користувач. Через них людина шукає, обирає й купує взуття. До таких модулів належать каталог товарів, кошик і оформлення замовлення. У цьому розділі буде розказано, як кожен з них працює, як зроблений і буде показаний приклади коду.

2.4.1 Модуль каталогу товарів

Каталог — це головне місце, де людина бачить усе доступне взуття. Тут можна фільтрувати товари (наприклад, за розміром або брендом), сортувати (наприклад, за ціною), перегортати сторінки, якщо товарів багато, і подивитися детальну інформацію про кожну пару (рис. 2.4.1)(рис. 2.4.2)(рис. 2.4.3

Ключові можливості каталогу:

- Фільтрація за брендом, категорією, розміром
- Сортування за ціною та назвою
- Пошук за ключовими словами
- Пагінація списку товарів

```
# Сортування
if sort == 'price_asc':
    shoes = shoes.order_by('price')
elif sort == 'price_desc':
    shoes = shoes.order_by('-price')
elif sort == 'name_asc':
    shoes = shoes.order_by('name')
elif sort == 'name_desc':
    shoes = shoes.order_by('-name')

paginator = Paginator(shoes, 6)
page_obj = paginator.get_page(page_number)
```

Рисунок 2.4.1 - Фрагемт коду(каталог товарів у views.py)

2.4.2 Модуль кошика покупок

Кошик працює навіть без входу в акаунт. Коли користувач додає товар, сайт створює запис у сесії — тобто тимчасово запам’ятовує, що саме вибрали і скільки штук. Усе зберігається у вигляді простого списку з ID товару і кількістю.

Основні функції:

- Додавання, зміна кількості, видалення товарів
- Відображення загальної вартості
- Підрахунок знижки

```python
@login_required
def add_to_cart(request, pk):
    cart = request.session.get('cart', {})
    print("Cart before:", cart)
    cart[str(pk)] = cart.get(str(pk), 0) + 1
    request.session['cart'] = cart
    print("Cart after:", cart)
    return redirect('cart')
```

Рисунок 2.4.2. - Фрагмент коду (додавання товару до кошика)

```html
    <form method="post">
      {% csrf_token %}
      <input type="hidden" name="add_to_cart" value="1">
      <button type="submit" class="btn btn-success mt-2">🛒 Додати до кошика</button>
    </form>
  </div>
</div>
```

Рисунок 2.4.3 - Фрагмент HTML (detail.html)

2.4.3 Модуль оформлення замовлення

Коли кошик заповнений, користувач переходить до оформлення замовлення. Там він вводить своє ім’я, адресу доставки та підтверджує покупку. Після цього замовлення надсилається в систему (рис. 2.4.4)

Основні етапи оформлення:

- Збір інформації користувача
- Збереження замовлення та зв'язок з товарами
- Генерація листа-підтвердження

```python
@login_required
def checkout(request):
    cart = request.session.get('cart', {})
    shoes = Shoe.objects.filter(id__in=cart.keys())
    total_price = sum(shoe.price * quantity for shoe, quantity in zip(shoes, cart.values()))

    if request.method == 'POST':
        name = request.POST.get('name')
        address = request.POST.get('address')

        if request.user.is_authenticated:
            order = Order.objects.create(
                user=request.user,
                name=name,
                address=address,
                status='processed'
            )

            for shoe in shoes:
                quantity = cart.get(str(shoe.id), 0)
                OrderItem.objects.create(order=order, shoe=shoe, quantity=quantity)

            # Email confirmation
            subject = 'Підтвердження замовлення'
            message = (
                f'Дякуємо, {request.user.username}! Ваше замовлення №{order.id} прийнято.\n\n'
                f'Доставка на адресу: {order.address}\n'
                f'Сума: {total_price} грн\n\nМи повідомимо, коли замовлення буде відправлено.'
            )

            send_mail(
                subject,
                message,
                'your_email@gmail.com',
                [request.user.email],
                fail_silently=False,
            )

            # Очистити кошик
            request.session['cart'] = {}
            return redirect('home')

    return render(request, 'store/checkout.html', {
        'shoes': shoes,
        'total_price': total_price
    })
```

Рисунок 2.4.4 - Фрагмент коду (створення замовлення)

Функціональні модулі — це основа зручного користування сайтом. Вони відповідають за те, щоб усе працювало чітко: вибір товарів, додавання в кошик, оформлення замовлення. Завдяки Django вдалося зробити все зрозуміло, просто і так, щоб було зручно і покупцям, і адміну сайту.

2.5 Інтеграція чат-бота для підтримки користувачів

Сьогодні, коли онлайн-магазини активно розвиваються, дуже важливо швидко й зручно спілкуватися з клієнтами. Один із кращих способів — додати чат-бота. Це розумна програма, яка сама відповідає на запитання, підказує з вибором взуття та допомагає оформити замовлення.

У цьому проєкті чат-бот зроблений як окрема частина сайту, яка підключена через спеціальні запити (REST API). Його головна задача — швидко відповідати на питання користувачів без участі живого оператора.

Архітектурно чат-бот складається з трьох ключових компонентів:

- Інтерфейс введення (форма або віджет на сайті)
- Обробник запитів на бекенді (view-функція в Django)
- Логіка відповіді: словникова база (rule-based) або інтеграція з OpenAI API (AI-based)

Спочатку бот був дуже простий — відповідав по заготовленому списку фраз. Потім його покращили: підключили OpenAI GPT, і тепер він може сам формувати відповіді, розуміючи суть запитання.

Бот працює так: користувач пише повідомлення в спеціальне поле на сайті. Сайт відправляє це повідомлення на сервер. Там Django обробляє текст і повертає відповідь. А відповідь знову показується людині на екрані. Усе це працює швидко й непомітно для користувача.

На сервері є спеціальна функція, яка називається chatbot_response. Вона приймає повідомлення від користувача (через запит) і повертає відповідь. У простій версії бот просто вибирає готову фразу з заздалегідь створеного списку.

Щоб бот відповідав розумно, його підключили до штучного інтелекту через OpenAI. Коли людина щось пише, сервер відправляє це питання в GPT. Модель читає текст, «думає» і повертає відповідь, яку бачить користувач.

Такий бот дуже зручний, бо може відповідати майже на будь-яке запитання — чи є товар, який розмір краще, або який колір підійде. Це можливо завдяки підключенню до GPT — розумного штучного інтелекту.

Щоб усе працювало, треба взяти спеціальний ключ із сайту OpenAI і вставити його в налаштування сайту. Це займає кілька хвилин.

Щоб ніхто не «спамив» або не писав поганих слів, можна обмежити кількість повідомлень і фільтрувати небажаний текст.

Бот може працювати не тільки на сайті, а й у месенджерах, як-от Telegram. Це відкриває більше можливостей для допомоги клієнтам.

Загалом, такий бот робить магазин набагато зручнішим. Він швидко відповідає, розвантажує підтримку і допомагає людям не чекати. І якщо треба — його можна легко розширити, наприклад, зробити голосову версію чи ще одного бота в іншому додатку.

2.6 Висновок

У цьому розділі було показано, як на практиці створюється зручний та сучасний онлайн-магазин взуття. Спочатку розробили загальну архітектуру сайту, щоб усе працювало швидко, безпечно і могло масштабуватись при потребі.

Базу даних спроєктували так, щоб вона зручно зберігала всю потрібну інформацію — про товари, користувачів, замовлення. Інтерфейс зробили простим, зрозумілим і приємним — з допомогою HTML, CSS і Bootstrap.

Усі головні функції — як-от каталог товарів, кошик, оформлення замовлення — реалізовані так, щоб користувач міг легко знайти і купити потрібне взуття. А інтеграція чат-бота дозволяє отримати відповіді на запитання просто на сайті, без очікувань.

Бот підключається до розумного сервісу OpenAI і вміє відповідати на складніші запити. Усе це робить платформу не тільки функціональною, а ще й готовою до майбутнього розвитку.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1 Розробка та впровадження веб-застосунку

Створення сайту — це найважливіша частина розробки онлайн-магазину. У цьому підрозділі розповідається, як саме був зроблений повноцінний магазин взуття на Django.

Тут реалізовано все необхідне: виведення товарів, робота з кошиком, оформлення замовлень, особисті кабінети користувачів та підтвердження покупки через email.

Основну увагу зосереджено на тому, як працює логіка на сервері (бекенд), як виглядають шаблони сторінок і як відбувається робота з базою даних.

Сервер сайту зроблений на Python за допомогою Django. Усі важливі дії — наприклад, перехід між сторінками або обробка форм — виконуються через спеціальні функції.

Система вміє обробляти відгуки, додавати товари в обране, створювати PDF-файли з деталями замовлення та надсилати підтвердження на email (рис. 3.1.1)(рис. 3.1.2)(рис. 3.1.3)(рис. 3.1.4)(рис. 3.1.5.)(рис. 3.1.6)(рис. 3.1.7)(рис. 3.1.8)

```python
from django.contrib.auth import login
from django.contrib.auth.decorators import login_required
from .models import Review
from django.http import HttpResponse
from django.template.loader import get_template
from xhtml2pdf import pisa
from django.contrib import messages
from django.core.mail import send_mail
from .forms import ShoeFilterForm
from django.core.paginator import Paginator



@login_required
def edit_review(request, pk):
    review = get_object_or_404(Review, pk=pk, user=request.user)

    if request.method == 'POST':
        review.text = request.POST.get('text')
        review.rating = int(request.POST.get('rating'))
        review.save()
        return redirect('detail', pk=review.shoe.pk)

    return render(request, 'store/edit_review.html', {
        'review': review,
        'rating_choices': [1, 2, 3, 4, 5],  # Ось це обов'язково!
    })

@login_required
def delete_review(request, pk):
    review = get_object_or_404(Review, pk=pk, user=request.user)
    shoe_id = review.shoe.pk
    review.delete()
    return redirect('detail', pk=shoe_id)

@login_required
def my_orders(request):
    orders = Order.objects.filter(user=request.user).order_by('-created_at')
    return render(request, 'store/my_orders.html', {'orders': orders})

def generate_pdf(order):
    template = get_template('store/invoice.html')
    html = template.render({'order': order})
    response = HttpResponse(content_type='application/pdf')
    response['Content-Disposition'] = f'attachment; filename="order_{order.id}.pdf"'
    pisa.CreatePDF(html, dest=response)
    return response
```

Рисунок 3.1.1 - Редагування відгуку та генерація PDF-файлу замовлення.

```python
def generate_pdf(order):
    response = HttpResponse(content_type='application/pdf')
    response['Content-Disposition'] = f'attachment; filename="order_{order.id}.pdf"'
    pisa.CreatePDF(html, dest=response)
    return response

def invoice_pdf(request, order_id):
    order = get_object_or_404(Order, pk=order_id)
    return generate_pdf(order)



@login_required
def add_to_favorites(request, pk):
    shoe = get_object_or_404(Shoe, pk=pk)
    Favorite.objects.get_or_create(user=request.user, shoe=shoe)
    return redirect(request.META.get('HTTP_REFERER', 'home'))

@login_required
def remove_from_favorites(request, pk):
    print("Товар видаляється з обраного")
    shoe = get_object_or_404(Shoe, pk=pk)
    Favorite.objects.filter(user=request.user, shoe=shoe).delete()
    return redirect(request.META.get('HTTP_REFERER', 'home'))

@login_required
def favorites_list(request):
    favorites = Favorite.objects.filter(user=request.user)
    return render(request, 'store/favorites.html', {'favorites': favorites})

def register(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('home')
    else:
        form = UserCreationForm()
    return render(request, 'store/register.html', {'form': form})

from django.shortcuts import render
from .models import Shoe, OrderItem, Order, Favorite

def home(request):
    category = request.GET.get('category')
    search = request.GET.get('search')
    sort = request.GET.get('sort')
```

Рисунок 3.1.2 - Функціонал додавання у вибране, перегляд вибраного.

```python
def home(request):



    banners = Banner.objects.filter(is_active=True).order_by('-created_at')
    shoes = Shoe.objects.all()

    # Фільтрація
    if form.is_valid():
        size = form.cleaned_data.get('size')
        brand = form.cleaned_data.get('brand')
        if size:
            shoes = shoes.filter(size=size)
        if brand:
            shoes = shoes.filter(brand=brand)
    if category:
        shoes = shoes.filter(category=category)
    if search:
        shoes = shoes.filter(name__icontains=search)
    if min_price:
        shoes = shoes.filter(price__gte=min_price)
    if max_price:
        shoes = shoes.filter(price__lte=max_price)

    # Сортування
    if sort == 'price_asc':
        shoes = shoes.order_by('price')
    elif sort == 'price_desc':
        shoes = shoes.order_by('-price')
    elif sort == 'name_asc':
        shoes = shoes.order_by('name')
    elif sort == 'name_desc':
        shoes = shoes.order_by('-name')

    paginator = Paginator(shoes, 6)
    page_obj = paginator.get_page(page_number)
    # Позначити, які товари в обраному
    if request.user.is_authenticated:
        favorites = Favorite.objects.filter(user=request.user).values_list('shoe_id', flat=True)
        for shoe in page_obj:
            shoe.is_favorite = shoe.id in favorites
    else:
        for shoe in shoes:
            shoe.is_favorite = False
            # Показувати 6 товарів на сторінку


    return render(request, 'store/home.html', {
```

Рисунок 3.1.3 - Обробка параметрів запитів користувача — фільтрація, сортування, пагінація.

```python
def detail(request, pk):
    shoe = get_object_or_404(Shoe, pk=pk)
    reviews = shoe.reviews.all().order_by('-created_at')

    if request.method == 'POST':
        # Додати до кошика
        if 'add_to_cart' in request.POST:
            cart = request.session.get('cart', {})
            cart[str(pk)] = cart.get(str(pk), 0) + 1
            request.session['cart'] = cart
            return redirect('cart')

        # Додати відгук
        elif 'add_review' in request.POST and request.user.is_authenticated:
            text = request.POST.get('text')
            rating = request.POST.get('rating')
            if text and rating:
                Review.objects.create(
                    shoe=shoe,
                    user=request.user,
                    text=text,
                    rating=rating
                )
                return redirect('detail', pk=shoe.pk)

    return render(request, 'store/detail.html', {
        'shoe': shoe,
        'reviews': reviews,
        'rating_choices': [1, 2, 3, 4, 5],
    })

def cart(request):
    cart = request.session.get('cart', {})
    shoes = []
    total = 0
    for pk, quantity in cart.items():
        shoe = get_object_or_404(Shoe, pk=pk)
        shoe.quantity = quantity
        shoe.total = shoe.price * quantity
        total += shoe.total
        shoes.append(shoe)
    return render(request, 'store/cart.html', {'cart_items': shoes, 'total': total})

from django.shortcuts import render, redirect, get_object_or_404
from .models import Shoe, Order, OrderItem
```

Рисунок 3.1.4 - Підрахунок загальної суми товарів у кошику.

```python
def detail(request, pk):
    })

def cart(request):
    cart = request.session.get('cart', {})
    shoes = []
    total = 0
    for pk, quantity in cart.items():
        shoe = get_object_or_404(Shoe, pk=pk)
        shoe.quantity = quantity
        shoe.total = shoe.price * quantity
        total += shoe.total
        shoes.append(shoe)
    return render(request, 'store/cart.html', {'cart_items': shoes, 'total': total})

from django.shortcuts import render, redirect, get_object_or_404
from .models import Shoe, Order, OrderItem

@login_required
def checkout(request):
    cart = request.session.get('cart', {})
    shoes = Shoe.objects.filter(id__in=cart.keys())
    total_price = sum(shoe.price * quantity for shoe, quantity in zip(shoes, cart.values()))

    if request.method == 'POST':
        name = request.POST.get('name')
        address = request.POST.get('address')

        if request.user.is_authenticated:
            order = Order.objects.create(
                user=request.user,
                name=name,
                address=address,
                status='processed'
            )

            for shoe in shoes:
                quantity = cart.get(str(shoe.id), 0)
                OrderItem.objects.create(order=order, shoe=shoe, quantity=quantity)

            # Email confirmation
            subject = 'Підтвердження замовлення'
            message = (
                f'Дякуємо, {request.user.username}! Ваше замовлення №{order.id} прийнято.\n\n'
                f'Доставка на адресу: {order.address}\n'
                f'Сума: {total_price} грн\n\nМи повідомимо, коли замовлення буде відправлено.'
            )

            send_mail(
```

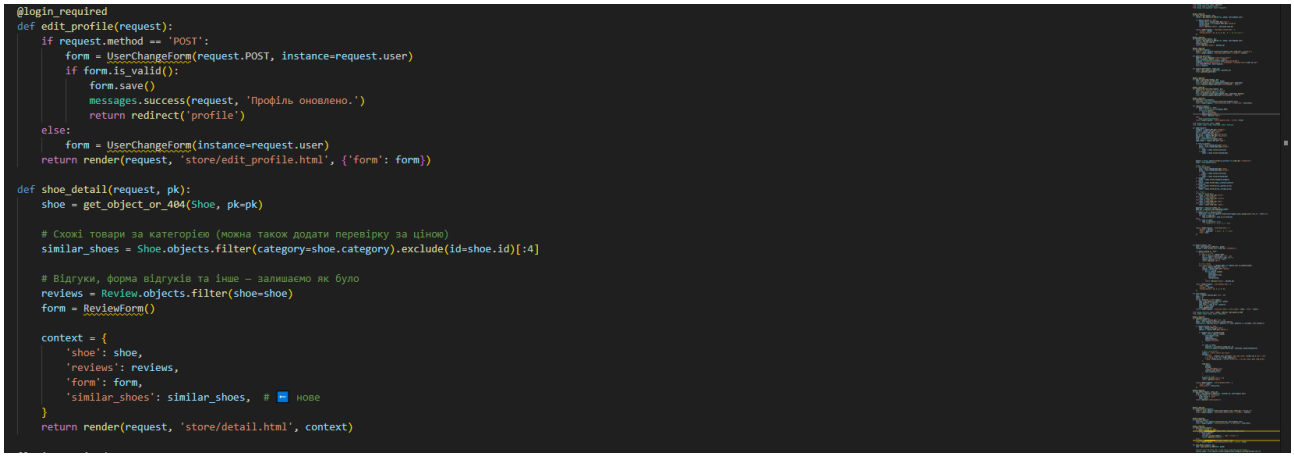
Рисунок 3.1.5 - Обробка замовлення, створення об'єктів Order та OrderItem.

```python
def checkout(request):
        if request.user.is_authenticated:
            order = Order.objects.create(
                user=request.user,
                name=name,
                address=address,
                status='processed'
            )

            for shoe in shoes:
                quantity = cart.get(str(shoe.id), 0)
                OrderItem.objects.create(order=order, shoe=shoe, quantity=quantity)

            # Email confirmation
            subject = 'Підтвердження замовлення'
            message = (
                f'Дякуємо, {request.user.username}! Ваше замовлення №{order.id} прийнято.\n\n'
                f'Доставка на адресу: {order.address}\n'
                f'Сума: {total_price} грн\n\nМи повідомимо, коли замовлення буде відправлено.'
            )

            send_mail(
                subject,
                message,
                'your_email@gmail.com',
                [request.user.email],
                fail_silently=False,
            )

            # Очистити кошик
            request.session['cart'] = {}
            return redirect('home')

    return render(request, 'store/checkout.html', {
        'shoes': shoes,
        'total_price': total_price
    })

@login_required
def pay_order(request, order_id):
    order = get_object_or_404(Order, id=order_id, user=request.user)
    if order.status == 'processed':
        order.status = 'paid'
        order.save()
    return redirect('order_history')
```

Рисунок 3.1.6 - Відправлення листа-підтвердження з інформацією про замовлення.

```python


@login_required
def profile(request):
    favorites = Shoe.objects.filter(favorite__user=request.user)
    return render(request, 'store/profile.html', {'favorites': favorites})

@login_required
def edit_profile(request):
    if request.method == 'POST':
        form = UserChangeForm(request.POST, instance=request.user)
        if form.is_valid():
            form.save()
            messages.success(request, 'Профіль оновлено.')
            return redirect('profile')
    else:
        form = UserChangeForm(instance=request.user)
    return render(request, 'store/edit_profile.html', {'form': form})

def shoe_detail(request, pk):
    shoe = get_object_or_404(Shoe, pk=pk)

    # Схожі товари за категорією (можна також додати перевірку за ціною)
    similar_shoes = Shoe.objects.filter(category=shoe.category).exclude(id=shoe.id)[:4]

    # Відгуки, форма відгуків та інше — залишаємо як було
    reviews = Review.objects.filter(shoe=shoe)
    form = ReviewForm()

    context = {
        'shoe': shoe,
        'reviews': reviews,
        'form': form,
        'similar_shoes': similar_shoes,  # ← нове
    }
    return render(request, 'store/detail.html', context)

```

Рисунок 3.1.7 - Обробка історії замовлень та перегляд профілю користувача.



```python
@login_required
def edit_profile(request):
    if request.method == 'POST':
        form = UserChangeForm(request.POST, instance=request.user)
        if form.is_valid():
            form.save()
            messages.success(request, 'Профіль оновлено.')
            return redirect('profile')
    else:
        form = UserChangeForm(instance=request.user)
    return render(request, 'store/edit_profile.html', {'form': form})

def shoe_detail(request, pk):
    shoe = get_object_or_404(Shoe, pk=pk)

    # Схожі товари за категорією (можна також додати перевірку за ціною)
    similar_shoes = Shoe.objects.filter(category=shoe.category).exclude(id=shoe.id)[:4]

    # Відгуки, форма відгуків та інше — залишаємо як було
    reviews = Review.objects.filter(shoe=shoe)
    form = ReviewForm()

    context = {
        'shoe': shoe,
        'reviews': reviews,
        'form': form,
        'similar_shoes': similar_shoes,  # ← нове
    }
    return render(request, 'store/detail.html', context)
```

Рисунок 3.1.8 - Вивід схожих товарів на сторінці товару та форма редагування профілю.

Загалом, веб-сайт працює так, як очікує звичайний користувач: усе починається з перегляду товару, потім людина може додати його до кошика, оформити замовлення й одразу отримати підтвердження на email. Усі ці дії відбуваються швидко й без збоїв, а сама система зберігає історію кожної покупки — це зручно і для користувача, і для адміністратора.

Інтерфейс сайту зроблений зрозумілим і простим, щоб будь-хто — навіть без технічного досвіду — міг легко знайти потрібне взуття, додати його в кошик або в обране. Усе виглядає охайно і добре працює на різних пристроях, включно з телефонами.

Усередині система працює на Django — це фреймворк, який дозволяє швидко реалізувати всі важливі функції: обробку форм, роботу з базою даних, перевірку безпеки, надсилання листів, тощо. Завдяки цьому можна не тільки підтримувати стабільну роботу сайту, а й легко розширювати його — додавати нові розділи, можливості чи інтегрувати платіжні сервіси.

Таким чином, розроблений веб-магазин є повноцінним онлайн-застосунком, який не тільки зручно використовувати, а ще й просто підтримувати та розвивати у майбутньому.

3.2 Тестування функціоналу та усунення помилок

Тестування — це обов’язковий етап у створенні будь-якого сайту, бо саме воно допомагає знайти й виправити помилки до того, як користувачі почнуть ним користуватись. Під час розробки онлайн-магазину взуття велика увага

приділялась перевірці того, як працює кошик і обране — тобто ті частини сайту, якими користуються найчастіше.

Перевірявся сайт вручну — так, як би ним користувалась звичайна людина. Натискались кнопки, додавались товари, переходили між сторінками, щоб упевнитись, що все працює правильно.

Одразу було вирішено протестувати кошик, бо це одна з головних частин магазину. Він має додавати товар, рахувати загальну суму і показувати все, що вибрав користувач. Але вже на початку з’ясувалось, що є проблема — після натискання кнопки «Додати до кошика» товар туди не потрапляв. Через це замовлення оформити було неможливо.

Після перевірки стало зрозуміло, що причина помилки була в тому, як оброблявся запит на додавання товару. У функції, яка відповідає за сторінку з товаром, неправильно працювала логіка — або не створювався кошик у сесії, або неправильно зберігався товар.

Код був змінений так, щоб сайт спочатку перевіряв, чи є вже кошик, і якщо його немає — створював. Також додалась можливість збільшувати кількість товару, якщо він вже є в кошику. Після цих змін усе знову протестувалось — і кошик почав працювати правильно.

Також перевірявся розділ із "обраним". Його суть у тому, що користувач може додати взуття до списку вподобаних товарів, щоб легко знайти його пізніше. Спочатку кнопка "обране" не працювала — після натискання нічого не відбувалося.

Причиною виявилась відсутність коду, який зв’язує користувача з товаром у базі даних. Щоб усе працювало правильно, були створені окремі функції для додавання й видалення з обраного. Вони записують інформацію в таблицю, де зберігається, хто і який товар додав. Після цього система почала працювати як треба.

Кожен модуль перевірявся не тільки на логіку роботи, а й на технічні помилки. Під час тестування особливу увагу звертали на такі речі:

- чи правильно передаються змінні з бекенду в шаблони;

- чи всі посилання на сайті відповідають маршрутам у файлі urls.py;
- чи дані з форм правильно відповідають тим, що зберігаються в моделях;
- чи справді інформація потрапляє в базу даних після дій користувача.

Під час тестування також було помічено, що на сторінці товару іноді неправильно відображалась знижка. Навіть коли знижка була 0%, все одно показувалась перекреслена ціна, ніби товар продається зі знижкою. Щоб це виправити, у шаблоні додалась перевірка: блок зі старою ціною з'являється лише тоді, коли знижка справді більша за 0%.

Ще одна частина тестування стосувалась відгуків. Перевірялась робота додавання, редагування та видалення коментарів. Було помічено, що якщо користувач залишав відгук без тексту або ставив неправильний рейтинг, система просто нічого не робила — не показувала помилку й не зберігала коментар. Цю проблему виправили: тепер обидва поля — текст і оцінка — є обов'язковими, а рейтинг має бути тільки в межах від 1 до 5.

Завдяки тестуванню вдалося заздалегідь виявити й виправити проблеми в роботі сайту. Це допомогло зробити платформу зручною, надійною та зрозумілою для користувачів. Коли все працює як треба, люди охочіше користуються сайтом і більше довіряють магазину.

3.3 Забезпечення інформаційної безпеки

Захист даних — дуже важлива частина під час створення будь-якого сайту, особливо якщо він пов'язаний з покупками. Онлайн-магазин взуття з чат-ботом зберігає особисту інформацію клієнтів, тому треба подбати про безпеку. У цьому розділі розглянемо, як саме в системі реалізовано захист даних та які методи використовуються для запобігання витокам чи зламам.

Передусім система має бути захищена від найпоширеніших небезпек в інтернеті. Серед них — спроби вкрасти дані через підставні запити до бази (SQL-ін'єкції), впровадження шкідливого коду на сторінки (XSS), несанкціоновані запити від імені користувача (CSRF) та перехоплення даних між клієнтом і сервером (атаки типу «людина посередині»).

Щоб сайт був безпечним, у ньому реалізовано кілька важливих речей.

По-перше, система не дозволяє вставляти шкідливі коди в поля для відгуків чи повідомлень. Навіть якщо хтось спробує — сайт це «очищає» і просто покаже як звичайний текст.

По-друге, усі форми на сайті мають спеціальний захисний код (токен), який не дає змоги підробити запит з іншого сайту. Це захищає від небажаних дій замість користувача.

Сайт працює через захищене з’єднання — якщо коротко, усе, що користувач надсилає чи отримує, шифрується. Це робиться через протокол HTTPS і спеціальний сертифікат.

Паролі не зберігаються як текст — вони зашифровані. Тобто навіть якщо хтось отримає доступ до бази, він не дізнається паролі.

Увійти в адмінку можуть тільки ті, хто має на це дозвіл. Є розділення прав: один може лише дивитись, інший — змінювати.

Сайт також «веде щоденник» — записує важливі події, як-от спроби злому, зміну пароля чи підозрілі дії. Це допомагає швидко реагувати на проблеми.

І ще: якщо користувач довго нічого не робить, система автоматично його «викидає» з акаунта. Це потрібно, щоб хтось сторонній не скористався відкритою сторінкою.

Також дуже важливо мати резервні копії. У системі налаштовано автоматичне збереження копій бази даних через певні проміжки часу. Це означає, що якщо щось станеться — наприклад, збій або втрата даних — все можна буде швидко відновити.

3.4 Результати робити чат-бота на базі AI

Інтеграція чат-бота зі штучним інтелектом у платформу магазину взуття суттєво покращила спілкування з користувачами. Раніше для отримання відповіді клієнту потрібно було чекати на оператора або шукати інформацію самостійно. Тепер усе набагато простіше: бот доступний у будь-який час і швидко відповідає на найпоширеніші питання — наприклад, про наявність товару, способи доставки чи повернення.

Завдяки цьому навантаження на службу підтримки значно зменшилося. Оператори більше не витрачають час на типові питання й можуть зосередитися на складніших запитах.

Користувачам це також сподобалося — вони стали частіше користуватись платформою, бо знають, що завжди можуть швидко отримати допомогу. Бот працює цілодобово, не потребує перерв і відповідає без затримок. Це створює позитивне враження про сервіс і підвищує довіру до магазину.

Таким чином, впровадження чат-бота зробило сервіс більш сучасним, зручним і ефективним як для клієнтів, так і для самої команди проекту.

Чат-бот — це розумна програма, яка може розпізнавати питання користувача та відповідати на них у режимі реального часу. У цьому проекті він працює на основі штучного інтелекту від OpenAI. Це дає змогу не лише відповідати на типові запити (наприклад, “Як зробити замовлення?”), а й розуміти більш складні або довільні питання, які користувач формулює своїми словами.

Основні функції чат-бота:

Надання консультацій щодо процесу замовлення;

Інформування про наявність товарів, знижки, новинки;

Пояснення умов доставки та оплати;

Надання контактних даних технічної підтримки;

Відповіді на загальні питання.

Чат-бот був спеціально налаштований на найпоширеніші запити, які зазвичай виникають у покупців онлайн-магазинів. Це стосується таких питань, як: “Чи є мій розмір?”, “Скільки триває доставка?”, “Як оплатити замовлення?” або “Як повернути товар?”. Завдяки цьому бот розуміє потреби клієнтів і дає точні відповіді без участі менеджера.

Один із великих плюсів полягає в тому, що чат-бот легко адаптується до змін. Наприклад, якщо в магазині оновлюється асортимент, змінюються умови доставки чи вводиться нова акція — бот може швидко “навчитися” цим змінам.

Для цього оновлюється його база знань або налаштування, і він починає враховувати нову інформацію у відповідях.

Це дозволяє зберігати актуальність та точність навіть при динамічних змінах у роботі магазину. Більше того, якщо аналіз показує, що користувачі часто ставлять певні нові питання — їх теж можна додати до сценаріїв. Таким чином, бот постійно розвивається разом із платформою й дедалі краще відповідає на потреби клієнтів.

Технічна реалізація

Чат-бот у проєкті реалізований як окремий модуль. Коли користувач пише йому повідомлення на сайті, ця інформація відправляється на сервер через JavaScript (через Ajax-запит, тобто без перезавантаження сторінки). Далі сервер обробляє повідомлення і надсилає його до OpenAI — штучного інтелекту, який формує відповідь. Потім ця відповідь повертається на сайт у форматі JSON і відразу з’являється у вікні чату.

Результати впровадження

Після впровадження чат-бота було проведено аналіз того, як саме користувачі взаємодіють із ним і наскільки він допомагає у повсякденному обслуговуванні клієнтів. Результати виявилися дуже позитивними.

По-перше, час відповіді на типові запитання скоротився до 1–3 секунд. Це означає, що користувачам не потрібно чекати на відповідь оператора — бот реагує майже миттєво, що значно підвищує зручність користування сайтом.

По-друге, кількість звернень до технічної підтримки зменшилась приблизно на 35%. Раніше багато користувачів писали або телефонували, щоб дізнатися, наприклад, про наявність товару, способи доставки чи повернення. Тепер ці питання вирішуються автоматично за допомогою бота.

Також, згідно з відгуками, понад 80% користувачів залишились задоволені якістю та швидкістю відповідей. Багато хто відзначив, що бот допомагає швидко знайти потрібну інформацію без зайвих дій.

Ще один плюс — це можливість зібрати статистику по найпопулярніших питаннях. Це дало змогу оновити розділ «Часті запитання» (FAQ) на сайті, а

також налаштувати бот так, щоб він ще краще розумів запити клієнтів і відповідав ще точніше.

Крім того, бот позитивно вплинув на стабільність платформи в години пік. Оскільки частина навантаження на підтримку та пошук інформації перейшла на нього, сайт працює швидше й не зависає, навіть коли багато людей заходять одночасно.

Аналіз та можливості покращення

AI-чат-бот, інтегрований у веб-платформу магазину, вже зараз демонструє високу ефективність у сфері онлайн-торгівлі. Але його потенціал далеко не вичерпаний. У майбутньому функціональність бота можна значно розширити, щоб ще більше покращити взаємодію з клієнтами.

Одним із наступних кроків може стати підтримка кількох мов. Наприклад, додавання англійської, польської чи інших мов дозволить обслуговувати ширшу аудиторію — це особливо актуально для бізнесів, які планують виходити на міжнародні ринки.

Також можна додати голосовий інтерфейс, який дасть змогу спілкуватися з ботом не лише через текст, а й голосом. Це зробить комунікацію ще зручнішою, особливо для людей, які користуються мобільними пристроями.

Ще один перспективний напрям — це інтеграція чат-бота з CRM-системами. Це дозволить ботові «знати» історію покупок клієнта, враховувати попередні запити та пропонувати індивідуальні поради або знижки.

Крім того, чат-бот можна навчити розуміти контекст попередніх розмов — тобто вдосконалити його здатність до самонавчання. Завдяки цьому відповіді стануть точнішими, а спілкування — більш природним.

І наостанок — цікава та перспективна функція, яка вже застосовується в деяких системах: аналіз емоційного тону повідомлень користувача. Це дасть змогу визначати, чи користувач задоволений, роздратований або розгублений — і відповідно змінювати стиль відповідей або сповіщати оператора про потребу втручання.

Порівняння до і після впровадження

До впровадження чат-бота уся взаємодія з клієнтами відбувалася вручну — або через електронну пошту, або по телефону. В таких умовах користувачам доводилося чекати на відповідь від кількох годин до навіть цілого дня, що часто призводило до невдоволення або втрати потенційних покупців.

Із появою чат-бота ситуація суттєво змінилася. Бот бере на себе відповідь на найпоширеніші запити — наприклад, щодо наявності товару, умов доставки чи повернення. Це дозволило скоротити час реагування буквально до кількох секунд. Користувач одразу отримує потрібну інформацію, без очікування чи залучення оператора, а служба підтримки може зосередитися на складніших питаннях.

Завдяки такій автоматизації сервіс став не лише швидшим, а й більш зручним та доступним у будь-який час доби (табл. 3.4)

Таблиця 3.4 – Зрівняння роботи онлайн-магазину до впровадження чат-бота і після

Параметр	До чат-бота	Після впровадження
Середній час відповіді	4 години	3 секунди
Кількість звернень на оператора	100%	65%
Витрати на підтримку	Високі	Знижені
Ступінь задоволеності	Середній	Високий

Реалізація чат-бота справді стала важливим кроком для розвитку платформи. Вона не тільки значно підвищила швидкість і якість обслуговування клієнтів, а й дозволила оптимізувати внутрішні ресурси — зменшити навантаження на операторів і скоротити витрати часу на стандартні запити.

Крім того, чат-бот створив зручний, сучасний канал комунікації, доступний 24/7. Це особливо важливо для масштабування проєкту, адже із

зростанням кількості користувачів зростають і обсяги звернень. Автоматизована підтримка дозволяє з легкістю обслуговувати велику аудиторію без втрати якості.

Таким чином, інтеграція чат-бота не лише покращила користувацький досвід, а й заклала фундамент для подальшого розвитку платформи.

3.5 Висновок

У цьому розділі йшлося про те, як створювався сайт магазину взуття та як його перевіряли перед запуском. В результаті вийшов зручний, зрозумілий і сучасний сайт, яким легко користуватися.

Під час роботи налаштовували все "під капотом": як зберігаються товари, як працює кошик, як оформлюється замовлення, як виглядає сайт на телефоні й комп’ютері. Особливо важливо було зробити так, щоб сайт працював стабільно та швидко.

Всі ключові частини — кошик, замовлення, обране, відгуки — перевіряли вручну. Знаходили баги, виправляли їх і ще раз тестували. Це дало змогу уникнути помилок до того, як сайт потрапив до реальних користувачів.

Також подбали про безпеку: щоб ніхто не зміг вкрасти дані, зламати адмінку чи зробити щось шкідливе. Усі дані передаються захищено, паролі зберігаються надійно, адмінку бачать тільки дозволені люди.

Окремо додали розумного чат-бота — він відповідає на запитання, допомагає з вибором товару, і все це в реальному часі. Тобто вже не потрібно чекати відповіді на пошту або дзвінка — все швидко і зручно. Завдяки йому зменшилось навантаження на підтримку, а клієнти стали задоволеніші.

У підсумку — сайт готовий до використання. Він зручний, працює як треба, захищений і має всі функції, щоб бути справжнім інтернет-магазином. Чітка структура, продумана логіка й автоматизація допомогли зробити його таким, яким він є зараз.

4 ОЦІНКА ЕФЕКТИВНОСТІ ТА ПЕРСПЕКТИВИ РОЗВИТКУ

4.1 Оцінка ефективності реалізованої системи

Сьогодні онлайн-магазини — це не просто зручний спосіб робити покупки, а й важлива частина успішного бізнесу. Створення магазину взуття з чат-ботом, який допомагає покупцям у реальному часі, дозволяє зробити обслуговування швидким і приємним. Люди можуть швидше знайти потрібне, задати запитання та отримати відповідь без очікування. У цьому розділі розглядається, наскільки ефективно працює така система: як вона поводиться технічно, наскільки зручною є для користувачів і як виконує свої функції.

Оцінка ефективності проводиться за такими критеріями:

- Функціональна повнота**:**

У створеному онлайн-магазині працюють всі основні функції, які потрібні для зручних покупок: каталог товарів, кошик, оформлення замовлень, список обраного, вхід і реєстрація, а також керування контентом через адмінпанель. До того ж, інтегрований чат-бот допомагає покупцям у реальному часі. Усі ці частини були перевірені під час тестування — і працюють стабільно.

- Зручність використання:

Інтерфейс сайту простий і зрозумілий — ним легко користуватись навіть на телефоні. Щоб знайти потрібне, вистачає кількох кліків. Завдяки Bootstrap сайт виглядає сучасно, а зручна структура сторінок допомагає швидко орієнтуватися в усьому, що на ньому є.

- Продуктивність системи:

Сайт справляється з багатьма користувачами одночасно без гальмувань. Django швидко обробляє запити від користувачів, а база даних PostgreSQL добре працює навіть тоді, коли в ній багато інформації.

- Надійність та безпека:

Сайт добре захищений. Від випадкових або зловмисних дій — типу спроб зламати систему чи вставити шкідливий код — все блокується. Кожен користувач бачить лише те, що йому дозволено. Якщо довго нічого не робити —

система автоматично виходить із акаунту. А всі замовлення й особисті дані зберігаються безпечно.

- Інтеграція AI-чат-бота:

Чат-бот відповідає на питання користувачів миттєво, що зменшує навантаження на працівників підтримки та покращує зручність користування сайтом. Він поєднує заздалегідь прописані відповіді з можливістю підключення до штучного інтелекту OpenAI, тому може давати точні та корисні відповіді навіть на складні або нестандартні запити.

- Зворотній зв’язок з користувачем:

Опитування користувачів, які тестували сайт, показало, що більшість залишились задоволені. Вони відзначили, що сайт простий у використанні, легко знайти потрібне, а чат-бот швидко і корисно відповідає на запитання.

- Потенціал до масштабування:

Платформа зроблена з розрахунком на майбутнє. Якщо потрібно — можна легко додати оплату карткою, розумні поради, сповіщення на телефон. Сайт витримає більше людей і більше товарів без проблем.

У результаті розробки було створено повноцінний онлайн-магазин взуття, який працює стабільно, зручно й ефективно. Усі ключові функції — перегляд товарів, додавання до кошика, оформлення замовлення, авторизація, відгуки, робота з обраним — працюють так, як очікує користувач. Інтерфейс простий і зрозумілий навіть для людей без досвіду покупок онлайн, а сайт добре виглядає і на комп’ютері, і на телефоні.

Одним з найважливіших рішень стало впровадження чат-бота з підтримкою штучного інтелекту. Завдяки цьому покупці можуть отримати відповідь на своє запитання за кілька секунд, не чекаючи листа або дзвінка. Це значно покращує якість обслуговування, знижує навантаження на операторів і підвищує загальне враження від магазину.

Крім цього, система побудована таким чином, що її легко розвивати далі. Можна додати онлайн-оплату, рекомендації товарів, push-сповіщення, нові модулі або адаптувати платформу до великої кількості відвідувачів.

Загалом, ця веб-платформа показала себе як сучасне, надійне рішення для електронної комерції, яке готове до масштабування й активного використання.

4.2. Порівняння з аналогами

У цьому підрозділі проведено простий і зрозумілий порівняльний аналіз створеної веб-платформи онлайн-магазину взуття з двома подібними рішеннями — *AnalogShop* та *eShoes*. Порівняння здійснювалося за основними показниками, які важливі як для користувачів, так і для власників бізнесу: зручність у користуванні, швидкодія, безпека, функціональність і адаптивність до різних пристроїв.

Порівняння показало, що власна платформа вигідно вирізняється на фоні конкурентів. Вона має сучасний зовнішній вигляд, просту й зрозумілу навігацію, стабільну роботу на мобільних і десктопних пристроях. Важливою відмінністю стало впровадження чат-бота — інструменту, який допомагає користувачам у реальному часі, не залучаючи операторів. Це суттєво покращує користувацький досвід.

Крім того, система демонструє високу стабільність, швидко реагує на дії користувача та надає розширені можливості адміністрування й масштабування. Порівняно з AnalogShop та eShoes, платформа виглядає більш сучасно, технологічно та зручно для клієнта.

Таким чином, розроблена система не лише відповідає вимогам сучасної електронної комерції, а й у багатьох аспектах випереджає існуючі рішення на ринку (табл. 4.2)

Таблиця 4.2 – Порівняльна таблиця

Критерій	Розроблена система	AnalogShop	eShoes
Наявність чат-бота	Так	Ні	Так
Мобільна адаптивність	Так	Так	Ні
Інтеграція з базою даних	Так	Так	Так
Можливість адміністрування	Так	Частково	Так
Рівень персоналізації	Високий	Середній	Низький
Швидкість завантаження сторінки	Швидка	Середній	Повільна
Безпека даних користувача	Високий	Середній	Низький

Як показує проведене порівняння, головною перевагою розробленої веб-платформи є інтегрований чат-бот на базі штучного інтелекту. Завдяки йому користувачі можуть швидко отримати відповіді на типові запитання — без очікування та участі живого оператора. Це не лише спрощує комунікацію, але й суттєво знижує навантаження на службу підтримки.

Ще одна важлива перевага — висока швидкість роботи сайту. У сучасному онлайн-середовищі користувачі очікують, що сторінка завантажиться за кілька секунд — і саме так працює ця система. Порівняно з іншими платформами, вона забезпечує стабільну та оперативну взаємодію навіть при великій кількості запитів.

Крім того, особливу увагу було приділено захисту особистих даних. Усі критично важливі елементи системи — авторизація, обробка замовлень, збереження інформації — працюють із дотриманням сучасних стандартів безпеки. Це дозволяє користувачам бути впевненими у збереженні своїх даних та комфорті під час користування платформою.

У сукупності ці фактори роблять платформу не просто конкурентоспроможною, а однією з найзручніших і технологічно просунутих серед подібних рішень.

4.3 Перспективи вдосконалення та масштабування проекту

Онлайн-магазин взуття з вбудованим чат-ботом уже зараз працює добре — все зручно, зрозуміло і швидко. Користувачі можуть легко знайти потрібне взуття, додати його в кошик, оформити замовлення, а чат-бот допомагає розібратися з питаннями.

Але світ не стоїть на місці. Люди очікують ще більшої зручності, швидшої реакції, нових можливостей. Тому платформу можна і далі розвивати — додати, наприклад, оплату онлайн, ще зручніші підказки від бота, повідомлення про акції, підтримку через месенджери або мобільний застосунок.

Це зробить магазин ще більш привабливим для покупців і допоможе йти в ногу з часом. Бо навіть хороша система може стати ще кращою, якщо вдосконалювати її крок за кроком.

- Розширення функціональних можливостей:

Одним із цікавих напрямів розвитку платформи є створення розумної системи персональних рекомендацій. Тобто сайт сам буде підказувати користувачеві, які товари можуть йому сподобатися — наприклад, на основі того, що він переглядав, що додавав у кошик або що раніше купував.

Це можна зробити за допомогою спеціальних алгоритмів машинного навчання, які “вчаться” на поведінці покупців. Завдяки цьому кожен бачитиме саме ті товари, які йому найбільше підходять. А це, своєю чергою, підвищує шанси, що людина зробить покупку — тобто збільшує прибуток магазину.

- Інтеграція з платіжними системами та службами доставки:

У майбутньому до сайту можна підключити зручні платіжні сервіси, такі як LiqPay, Portmone, Stripe або PayPal. Це дозволить покупцям оплачувати замовлення прямо на сайті, швидко й безпечно.

Крім того, є можливість автоматично зв’язати сайт із службами доставки — наприклад, Новою Поштою, Укрпоштою або Meest. Завдяки цьому система зможе сама створювати накладні, відстежувати посилки та повідомляти клієнтів, де знаходиться їхнє замовлення. Це зробить процес покупки ще зручнішим як для магазину, так і для покупця.

- Мобільна адаптація або мобільний застосунок:

Попри те, що сайт добре працює на різних пристроях, окремий мобільний додаток став би ще зручнішим способом користування сервісом. Такий застосунок дозволить покупцям швидше знаходити потрібне взуття, отримувати повідомлення про акції або статус замовлення прямо на телефон, а також мати доступ до деяких функцій навіть без інтернету. Це зробить платформу ще ближчою до своїх клієнтів.

- Вдосконалення чат-бота:

Наразі чат-бот добре справляється з основними питаннями від клієнтів. У майбутньому його можна зробити ще розумнішим — підключити новішу версію GPT або навчити на внутрішніх даних магазину. Це дозволить боту відповідати на складніші запити, допомагати з вибором товарів і навіть самостійно оформлювати замовлення. Крім того, цікавою ідеєю є додавання голосового управління — щоб користувачі могли просто говорити, а бот відповідав.

- Розширення асортименту та мультикатегорійність:

Зараз платформа зосереджена на продажу взуття, але в майбутньому її можна розширити до повноцінного інтернет-магазину з різними категоріями товарів. Це допоможе залучити більше покупців і збільшити прибуток. Щоб реалізувати таке масштабування, потрібно передбачити підтримку кількох товарних категорій, можливість додавати різні бренди, а також зручні фільтри й сортування для кожної групи товарів.

- Підвищення безпеки та продуктивності:

Коли користувачів стає більше, зростає й навантаження на сервер. Щоб платформа працювала стабільно, потрібно подбати про масштабування. Найзручніше це зробити за допомогою хмарних сервісів, як-от AWS, Google Cloud чи DigitalOcean — вони дозволяють автоматично збільшувати ресурси, коли це потрібно. Щоб сайт працював ще швидше, можна підключити кешування (наприклад, Redis або Memcached), а також використати CDN для пришвидшення завантаження сторінок. Не менш важливо — регулярно створювати резервні копії бази даних на кількох рівнях, щоб у разі збою все можна було легко відновити.

- Підтримка декількох мов та валют:

Щоб платформа могла зростати й далі, важливо зробити її зручною для людей з інших країн. Для цього можна додати перемикач мов і можливість вибору валюти. Тоді користувачі з-за кордону зможуть легко робити замовлення, а магазин зможе вийти на нові ринки й знаходити нових партнерів. Це природний і логічний крок для розвитку.

- Аналітика та BI-рішення:

Щоб краще розуміти, як працює онлайн-магазин і що саме цікавить покупців, важливо мати під рукою зручні аналітичні інструменти. Інтеграція з такими системами, як Google Analytics, Power BI або Tableau, дає змогу візуалізувати основні показники — наприклад, скільки людей переглядають товари, скільки з них роблять покупки, звідки вони приходять і який у них середній чек. Завдяки цьому власник платформи може швидко бачити, що працює добре, а що потребує змін, і приймати обґрунтовані рішення для розвитку бізнесу.

4.4 Висновок

У четвертому розділі було підбито підсумки реалізації веб-платформи онлайн-магазину взуття, яка об’єднує класичний функціонал електронної торгівлі з сучасними технологіями підтримки користувачів — зокрема, чат-ботом на основі штучного інтелекту. Платформа була створена з урахуванням

потреб сучасних користувачів: вона швидко працює, зручна у користуванні, має зрозумілу навігацію та адаптивний дизайн для різних пристроїв. Тестування показало, що всі модулі працюють стабільно, а чат-бот справляється зі своїми завданнями та покращує сервіс.

Порівняння з існуючими аналогами (наприклад, AnalogShop та eShoes) дало змогу побачити, що розроблена система нічим не поступається за функціоналом. Більше того — має кілька помітних переваг, таких як інтерактивний чат-бот, розширені фільтри для пошуку товарів, гнучка адмінпанель та грамотно побудована база даних. Ці фактори роблять платформу не лише зручною для покупців, а й простою в адмініструванні.

Також було оцінено економічну доцільність впровадження подібного рішення. Автоматизація процесів (наприклад, консультацій через чат-бот) допомагає зменшити навантаження на команду підтримки, а отже — і витрати. Інвестиції в розробку й запуск проекту є виправданими, адже створена система здатна забезпечити стабільний дохід при правильному просуванні.

У розділі також окреслено подальші кроки для розвитку платформи: підключення платіжних сервісів, створення мобільного додатку, розширення можливостей чат-бота, а також впровадження персоналізованих рекомендацій для користувачів. Усе це дозволить утримати конкурентну перевагу, реагувати на зміну попиту та відповідати очікуванням сучасного покупця.

Загалом, розроблена веб-платформа — це не просто навчальний проект, а повноцінний програмний продукт із реальним потенціалом для масштабування, комерційного використання й подальшого розвитку.

Висновки

У результаті виконання бакалаврської кваліфікаційної роботи було успішно створено сучасну веб-платформу для онлайн-продажу взуття з вбудованим чат-ботом, який працює на основі штучного інтелекту. Проект охопив усі основні етапи розробки — від ідеї та планування до реального впровадження, тестування та оцінки ефективності.

Перший розділ дозволив глибше зануритися в тему — було проаналізовано сучасні тренди у сфері онлайн-торгівлі, розглянуто популярні рішення та їхні обмеження. Це дало змогу визначити, які функції справді важливі для зручного інтернет-магазину та на що варто звернути увагу під час розробки.

У другому розділі здійснено проектування самої платформи — продумано структуру, реалізовано зручний інтерфейс для користувача, налаштовано роботу з базою даних та реалізовано всі ключові функції: перегляд товарів, фільтри, реєстрацію, кошик, замовлення тощо. Окремим і важливим елементом став чат-бот, який відповідає на запити користувачів у реальному часі й допомагає з навігацією та вибором товарів.

Третій розділ показав, як саме система була впроваджена й протестована. У процесі тестування вдалося виявити і виправити ряд помилок, що суттєво підвищило стабільність роботи. Також було реалізовано базові заходи безпеки — обмеження доступу, захист форм, шифрування тощо. У підсумку система стала не лише функціональною, а й безпечною для користувачів.

У четвертому розділі проведено оцінку ефективності — як з технічної точки зору, так і з точки зору зручності для користувачів. Було порівняно платформу з іншими схожими рішеннями, виявлено її сильні сторони, а також намічено напрямки для майбутнього розвитку: від мобільного застосунку до інтеграції з платіжними й логістичними сервісами.

У підсумку можна сказати, що поставлені цілі були досягнуті. Система вийшла повноцінною, зручною та готовою до використання як у навчальному, так і в реальному комерційному середовищі. Вона показує, що поєднання сучасних веб-технологій з можливостями штучного інтелекту здатне значно

покращити якість онлайн-сервісів і зробити покупку максимально комфортною для клієнта.

Перелік посилань

1. Django Software Foundation. Django Documentation. [Електронний ресурс]. – Режим доступу: https://docs.djangoproject.com
2. МакКі Дж. Django на прикладах. Практичний курс веб-розробки. – К.: Діалектика, 2021. – 384 с.
3. Грассманн Ф. Python. Повний курс програмування. – Львів: Видавництво «Новий Світ», 2022. – 560 с.
4. Чижов Д. Архітектура веб-застосунків. – Харків: Фоліо, 2021. – 312 с.
5. Nielsen J. Usability Engineering. – San Francisco: Morgan Kaufmann, 2012. – 362 p.
6. ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models.
7. Резнік А. Створення інтерфейсів з використанням Bootstrap 5. – К.: Техніка, 2021. – 228 с.
8. Sommerville I. Software Engineering. 10th Edition. – Pearson Education, 2015. – 792 p.
9. Stone D., Jarrett C. User Interface Design and Evaluation. – San Francisco: Morgan Kaufmann, 2005. – 608 p.
10. PostgreSQL Global Development Group. PostgreSQL Documentation. – Режим доступу: https://www.postgresql.org/docs/
11. Коломоєць А. Бази даних у проєктуванні інформаційних систем. – Харків: ХНУРЕ, 2020. – 308 с.
12. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. – Pearson, 2020. – 1152 p.
13. Назаренко Ю.Ш. Розробка інтелектуальних систем на базі Python. – Львів: Астролябія, 2022. – 416 с.

14. McTear M. Conversational AI: Dialogue Systems, Conversational Agents, and Chatbots. – Springer, 2020. – 345 p.

15. Сурков І.В. Кібербезпека веб-додатків. – Київ: КНУБА, 2020. – 240 с.

16. Шахов А. Електронна комерція: теорія і практика. – К.: КНЕУ, 2021. – 275 с.

17. Mitkov R. The Oxford Handbook of Computational Linguistics. – Oxford University Press, 2022. – 784 p.

18. Freeman A. Pro ASP.NET Core MVC. – Apress, 2020. – 1080 p.

19. Hussain F. Machine Learning for Web Applications. – Springer, 2021. – 245 p.

20. Глушков А.В. Інформаційні технології в системах підтримки прийняття рішень. – К.: Наука, 2020. – 312 с.

21. Кузнєцова Л.П. Психологія користувача в цифровому середовищі. – Харків: ВНТУ, 2022. – 198 с.

22. Гаврилюк А. Штучний інтелект в електронній торгівлі. – Львів: ЛьвДУФК, 2021. – 230 с.

23. GitHub Docs. GitHub Actions Documentation. [Електронний ресурс]. – Режим доступу: https://docs.github.com/en/actions

24. OpenAI. ChatGPT API Documentation. [Електронний ресурс]. – Режим доступу: https://platform.openai.com/docs/

25. Dede A. Web Security: A WhiteHat Perspective. – Wiley, 2021. – 310 p.

26. EU General Data Protection Regulation (GDPR). [Електронний ресурс]. – Режим доступу: https://gdpr.eu/

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

1

ВЕБ ПЛАТФОРМА ОНЛАЙН-МАГАЗИНУ ВЗУТТЯ З АВТОМАТИЗОВАНИМ ЧАТ-БОТОМ ДЛЯ ПІДТРИМКИ КОРИСТУВАЧІВ

2

Мета, об'єкт і предмет дослідження

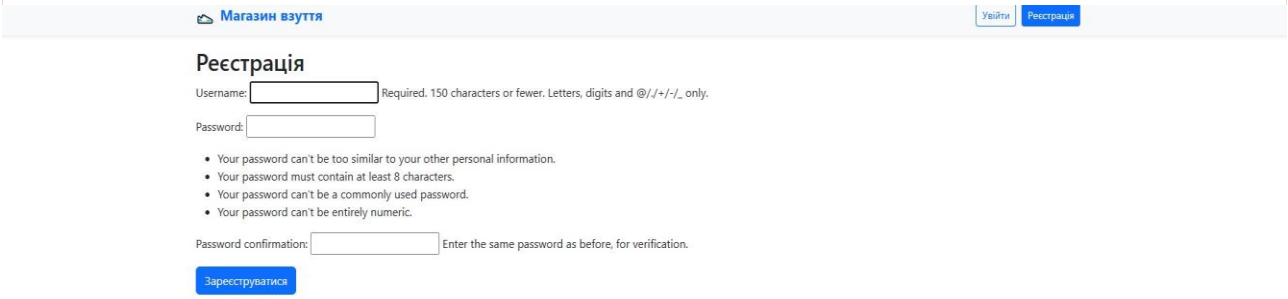
- Мета роботи:
- Створення функціонального, адаптивного вебзастосунку для онлайн-продажу взуття.

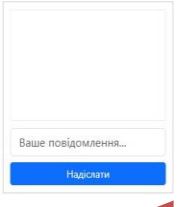
- Об'єкт дослідження:
- Процес розробки вебзастосунків у сфері електронної комерції.

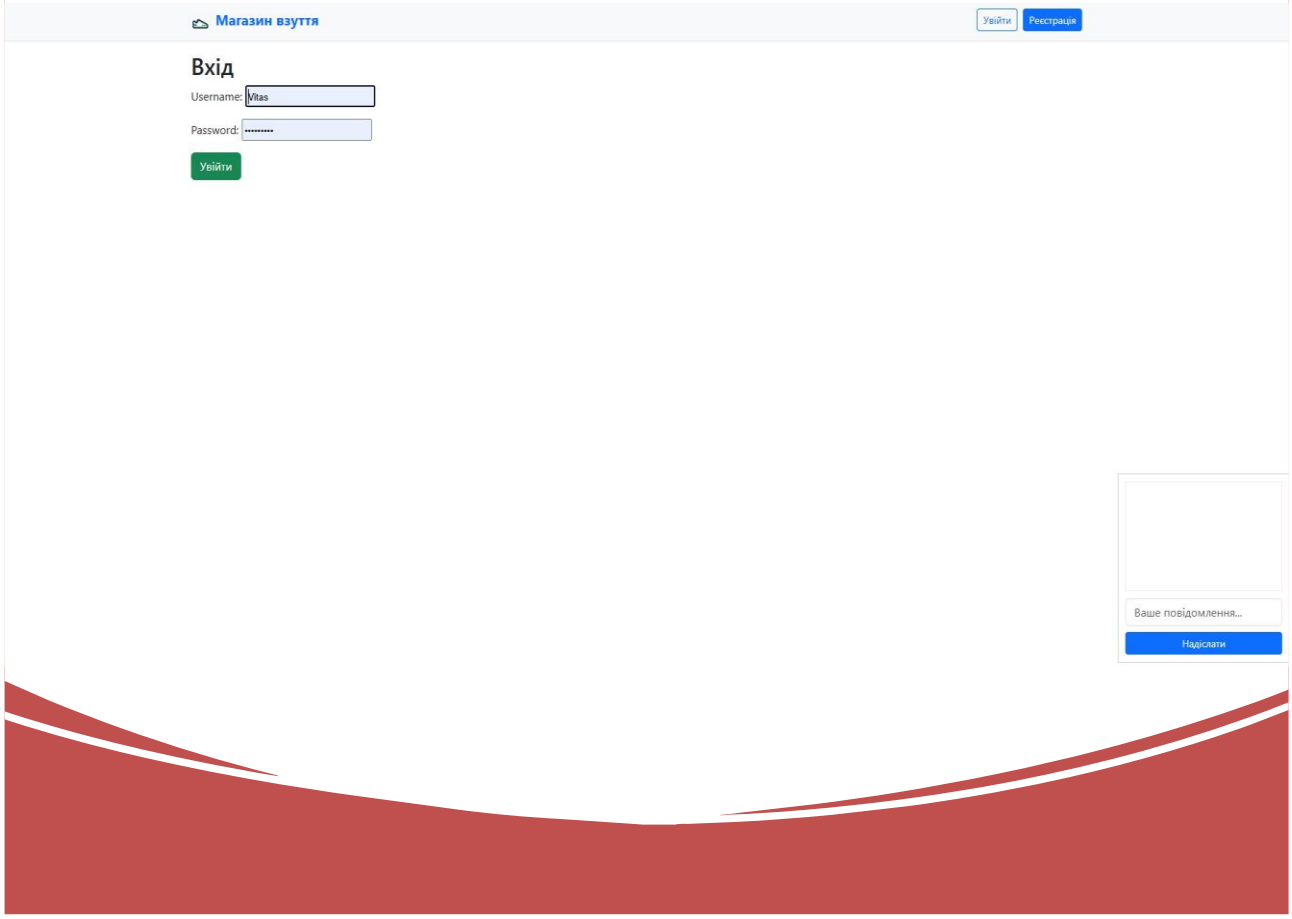
- Предмет дослідження:
- Технології створення інтернет-магазину за допомогою Python.

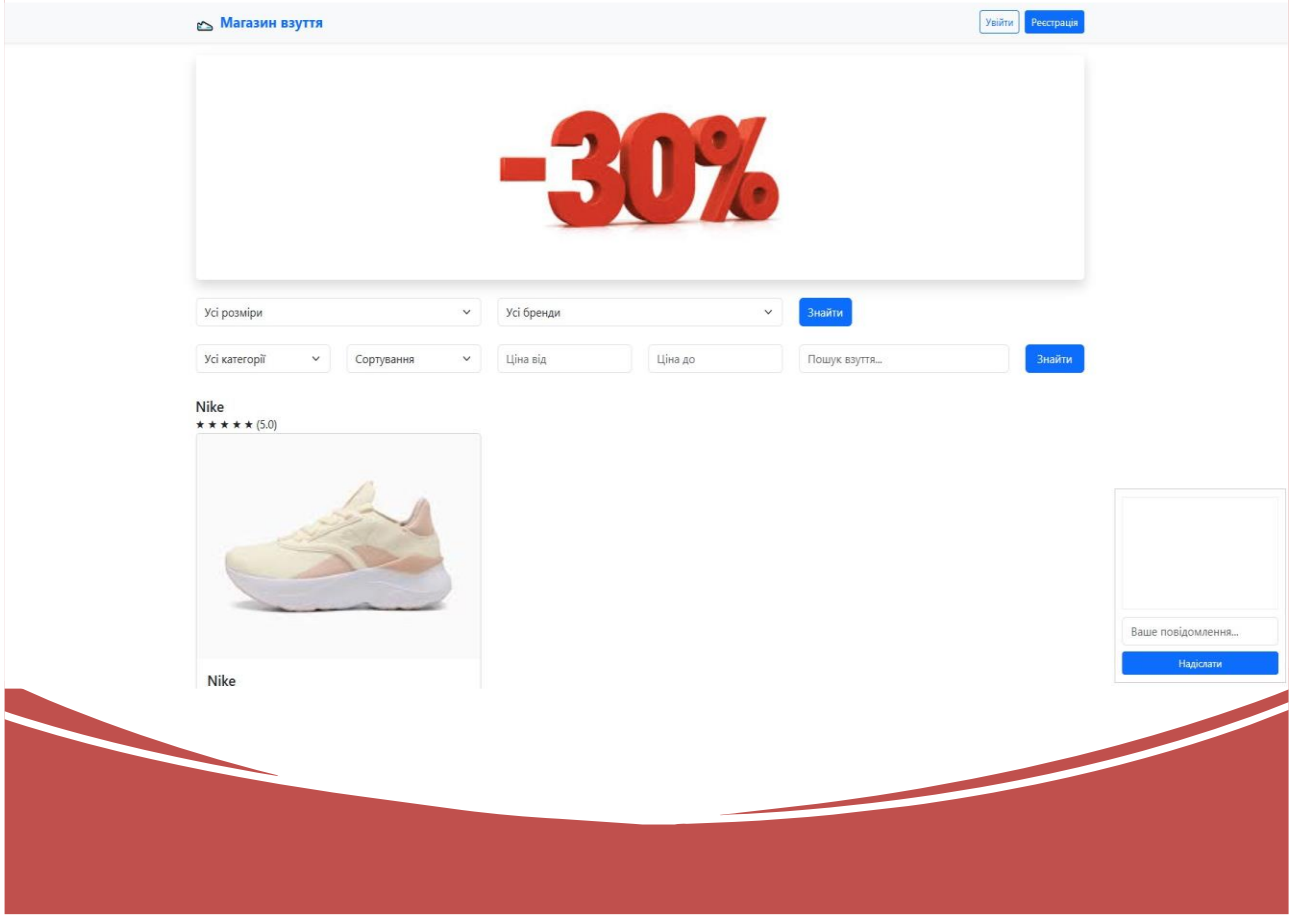
Основні функціональні можливості

- - Реєстрація, вхід, профіль користувача
- - Каталог товарів з фільтрацією
- - Детальна сторінка з рейтингами й відгуками
- - Обране / Кошик / Оформлення замовлення
- - Статуси замовлень (оброблено, відправлено, доставлено)
- - Email-підтвердження замовлення
- - Чат-бот
- - Адаптивна верстка (Bootstrap), пагінація
- - Адмінпанель






Магазин взуття
Увійти
Реєстрація
Вхід
Username:
Vitas
Password:
Увійти
Ваше повідомлення...
Надіслати


Магазин взуття
Увійти
Реєстрація
-30%
Усі розміри
Усі бренди
Знайти
Усі категорії
Сортування
Ціна від
Ціна до
Пошук взуття...
Знайти
Nike
★ ★ ★ ★ ★ (5.0)
Nike
Ваше повідомлення...
Надіслати

Магазин взуття

Увійти Реєстрація

Nike

Зручне взуття для бігу

~~3500.00 грн~~

2450.000 грн -30%

🛒 Додати до кошика

Відгуки

Vitas — 5★

17.05.2025 11:24

Nice=toP

Увійдіть, щоб залишити відгук.

Схожі товари

Немає схожих товарів.

Середній рейтинг: ★ ★ ★ ★ ★ (5.0)

Ваше повідомлення...

Надіслати

Магазин взуття

Увійти Реєстрація

Кошик

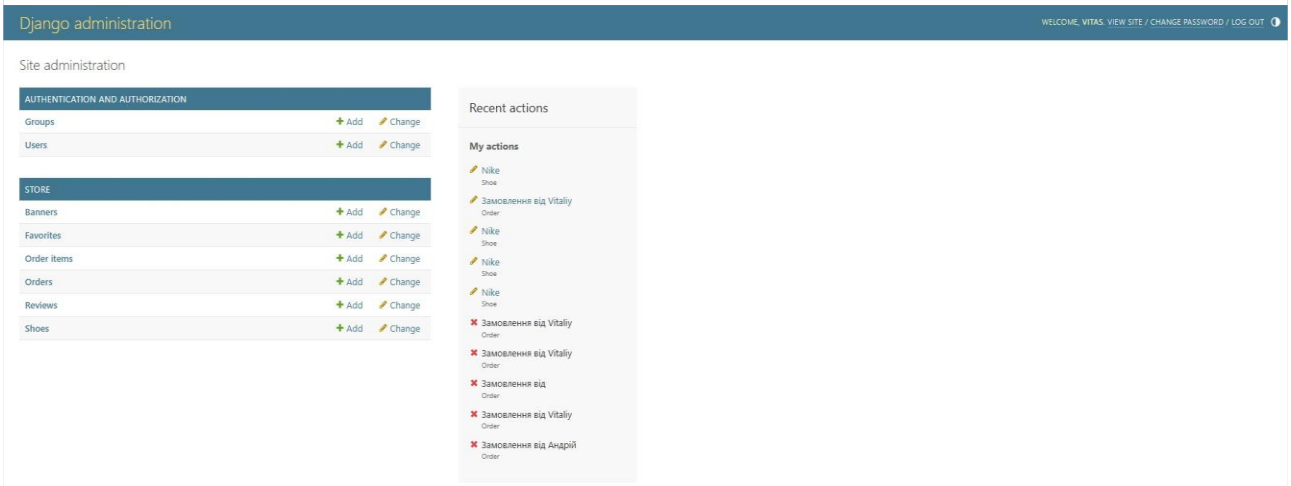
— 1 шт — 3500.00 грн Видалити

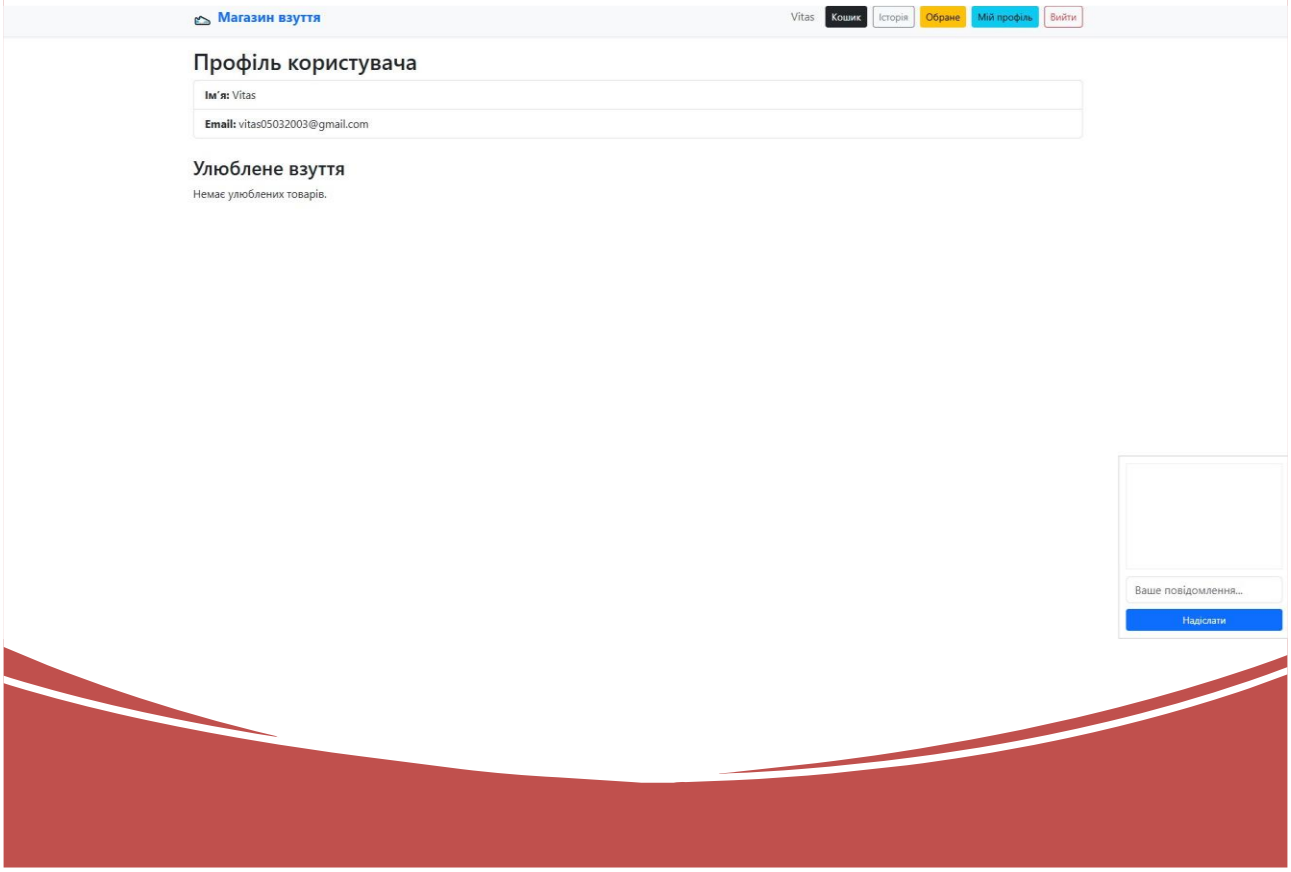
Разом: грн

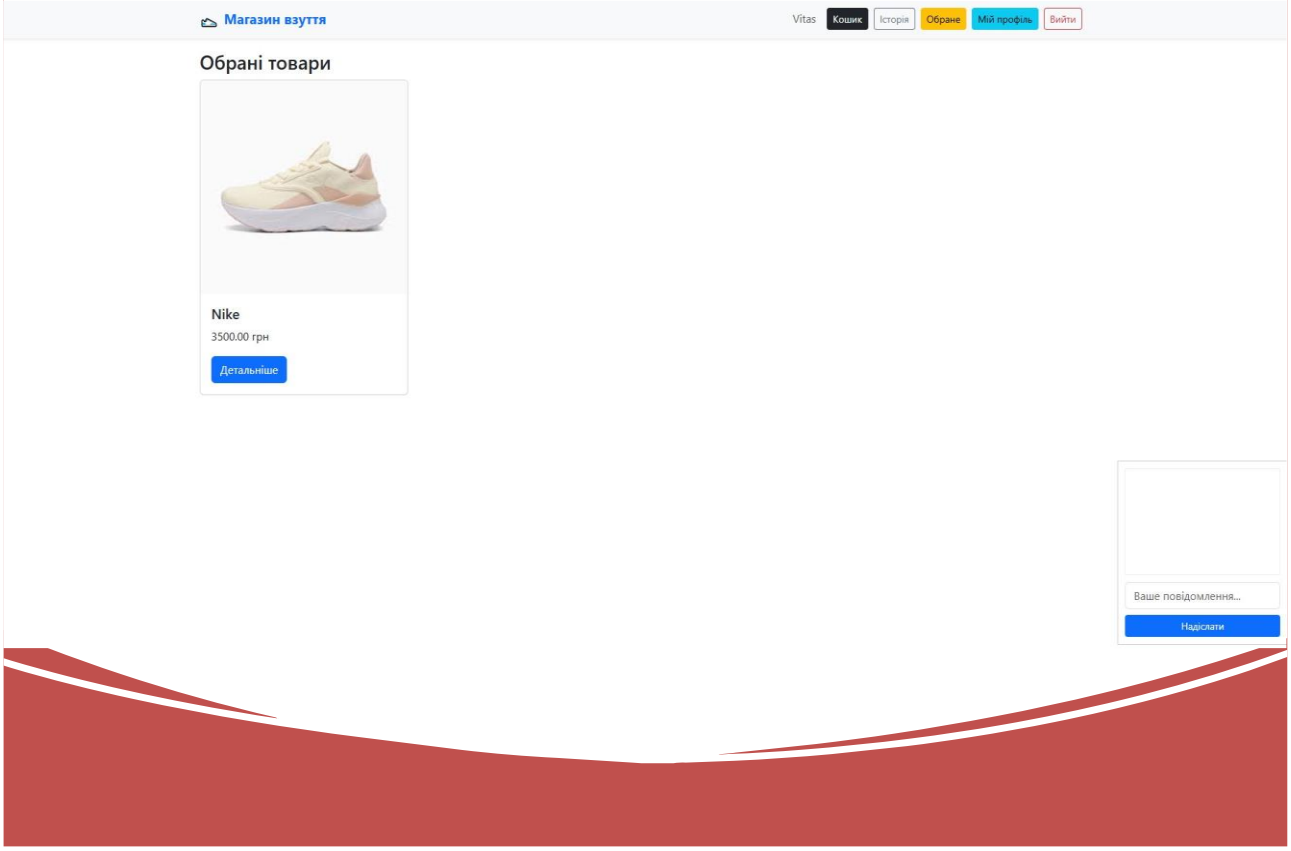
Оформити замовлення

Ваше повідомлення...

Надіслати


Django administration
WELCOME, VITAS. VIEW SITE / CHANGE PASSWORD / LOG OUT
Site administration
AUTHENTICATION AND AUTHORIZATION
Groups + Add Change
Users + Add Change
STORE
Banners + Add Change
Favorites + Add Change
Order items + Add Change
Orders + Add Change
Reviews + Add Change
Shoes + Add Change
Recent actions
My actions
Nike
Shoe
Замовлення від Vitaliy
Order
Nike
Shoe
Nike
Shoe
Nike
Shoe
Замовлення від Vitaliy
Order
Замовлення від Vitaliy
Order
Замовлення від
Order
Замовлення від Vitaliy
Order
Замовлення від Андрій
Order


Магазин взуття
Vitas
Кошик
Історія
Обране
Мій профіль
Вийти
Профіль користувача
Ім'я: Vitas
Email: vitas05032003@gmail.com
Улюблене взуття
Немає улюблених товарів.
Ваше повідомлення...
Надіслати



Результати роботи

- - Створено повноцінний інтернет-магазин взуття
- - Реалізовано зручний інтерфейс для користувачів і адміністратора
- - Забезпечено підтримку ключових функцій e-commerce
- - Додано можливості штучного інтелекту (чат-бот)