МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
(підпис)
червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВР

зі спеціальності 122 - Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Інтернет-магазин оригінального одягу та взуття»
здобувача групи ІН - 03 Надточій Едуарда Олександровича

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

Керівник,
доцент кафедри комп'ютерних наук,
кандидат фізико-математичних наук Надія ТИРКУСОВА
(підпис)

СУМИ – 2024

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-03 Надточій Едуарда Олександровича

1. Тема роботи: «Інтернет-магазин оригінального одягу та взуття»
затверджую наказом по СумДУ від *«22» квітня 2024 р. № 0414-IV*
2. Термін здачі здобувачем кваліфікаційної роботи *до 1 червня 2024 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для розробки інтернет-магазину. 3) Розробка інтернет-магазину для продажу оригінального одягу та взуття. 4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «06» травня 2024р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*	06.05.24 – 10.05.24	
2	*Огляд технологій, що використовуються для розробки інтернет-магазину*	11.05.24 – 17.05.24	
3	*Розробка інтернет-магазину для продажу оригінального одягу та взуття*	18.05.24 – 20.05.24	
4	*Аналіз отриманих результатів*	21.05.24 – 28.05.24	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	27.05.24 – 31.05.24	

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 67 стр., 44 рис., 2 додатки, 16 використаних джерел.

Обґрунтування актуальності теми роботи – тема кваліфікаційної роботи є актуальною, оскільки електронна комерція стає все більш популярною в сучасному світі. Користувачі мають дедалі більші очікування щодо якості та зручності веб-ресурсів, інтерес до моди, одягу та взуття швидко зростає.

Об’єкт дослідження — інтернет-магазин одягу та взуття.

Мета роботи — розробка інтернет-магазину оригінального одягу та взуття.

Методи дослідження — алгоритми створення веб сайтів з використанням певного стеку технологій.

Результати — розроблено інтернет-магазин для реалізації оригінального одягу та взуття, користувач має можливість передивитись список доступних товарів та придбати їх. Сайт створений на базі мови програмування Python з використанням фреймворку Django.

ІНТЕРНЕТ-МАГАЗИН, ОРИГІНАЛЬНИЙ ОДЯГ ТА ВЗУТТЯ, PYTHON, DJANGO, POSTGRESQL.

ЗМІСТ

ВСТУП

Актуальність. Тема кваліфікаційної роботи є актуальною, оскільки електронна комерція стає все більш популярною в сучасному світі. Користувачі мають дедалі більші очікування щодо якості та зручності веб-ресурсів, інтерес до моди, одягу та взуття швидко зростає.

Об’єкт дослідження. Процес продажу одягу та взуття.

Предмет дослідження. Методологія створення сайту та керування ним для продажу оригінального одягу та взуття.

Гіпотеза. Розроблений інтернет-магазин, який пропонує широкий вибір оригінального та стильного взуття та одягу за доступними цінами, буде успішним серед молодих людей, які шукають унікальні та модні речі.

Новизна. Розробка подібного сайту створить можливість допомогти людям знайти та придбати саме оригінальні та брендові речі за найкращою ціною.

Структура. Дана робота складається зі вступу, аналітичного огляду, постановки задачі, вибір методу розв’язання поставленої задачі, проектування інтернет-магазину, розробки інтернет-магазину, тестування, висновків та списку використаних джерел.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Інтернет-магазин оригінального взуття та одягу

Здається малоймовірно, що хтось звернеться до звичайного фізичного магазину, який знаходиться в іншому кінці міста, за винятком випадків, коли там продається щось дуже унікальне. В більшості випадків звичайні магазини відвідують мешканці або робітники, які живуть або працюють поблизу.

Тепер клієнтам не потрібно обходити кілька магазинів в пошуках товару за вигідною ціною. Вся необхідна інформація доступна на сайті, і це можна зробити, не виходячи з дому. Інтернет-магазини приймають замовлення цілодобово, без перерви на обід і вихідних, в зручний для клієнта час.

Інтернет-магазини можуть працювати на всю країну і навіть на весь світ, не обмежуючись конкретним регіоном. Це, безсумнівно, розширює коло потенційних покупців[2].

Для створення успішного інтернет-магазину потрібно дотримуватися кількох етапів:

1. Планування – на цьому етапі визначаються цілі та завдання сайту, аналізуються потреби цільової аудиторії, а також визначаються основні параметри, такі як тип сайту, платформа, функціональність, дизайн і контент.
2. Дизайн – розробляється дизайн сайту, який привертає увагу користувачів і полегшує пошук інформації. Дизайн повинен бути простим, та зручним у використанні.
3. Верстка та програмування – цей процес включає в себе об'єднання дизайну з функціональністю, що перетворює сайт в робочий інструмент.
4. Тестування – на цьому етапі перевіряються всі компоненти сайту, щоб переконатися, що вони працюють коректно і без помилок.
5. Запуск – фінальний етап створення веб-сайту. Після запуску сайту потрібно регулярно моніторити його роботу і оновлювати функціонал[3].

1.2 Аналіз конкурентів

Змагання між інтернет-магазинами ніколи не припиняється, оскільки підприємці постійно борються за трафік і клієнтів, щоб будувати успішний бізнес. Важливо мати глибоке розуміння своїх конкурентів, їхніх переваг і недоліків. Аналіз конкурентів допомагає здобути цю інформацію та використовувати її для власного розвитку[4]. Для цього аналізу були обрані два інтернет-магазини:

Rozetka – це один з найбільш відомих торгових ресурсів України, який має великий асортимент товарів, таких як електроніка, побутова техніка, автотовари, косметика, книги та одяг (рис. 1.1). Серед переваг цього сайту можна відзначити привабливий дизайн, зручну навігацію та можливість сортування товарів, а також швидку доставку. Проте серед недоліків можна вказати час відповіді сервісного центру та проблеми з гарантією[5].

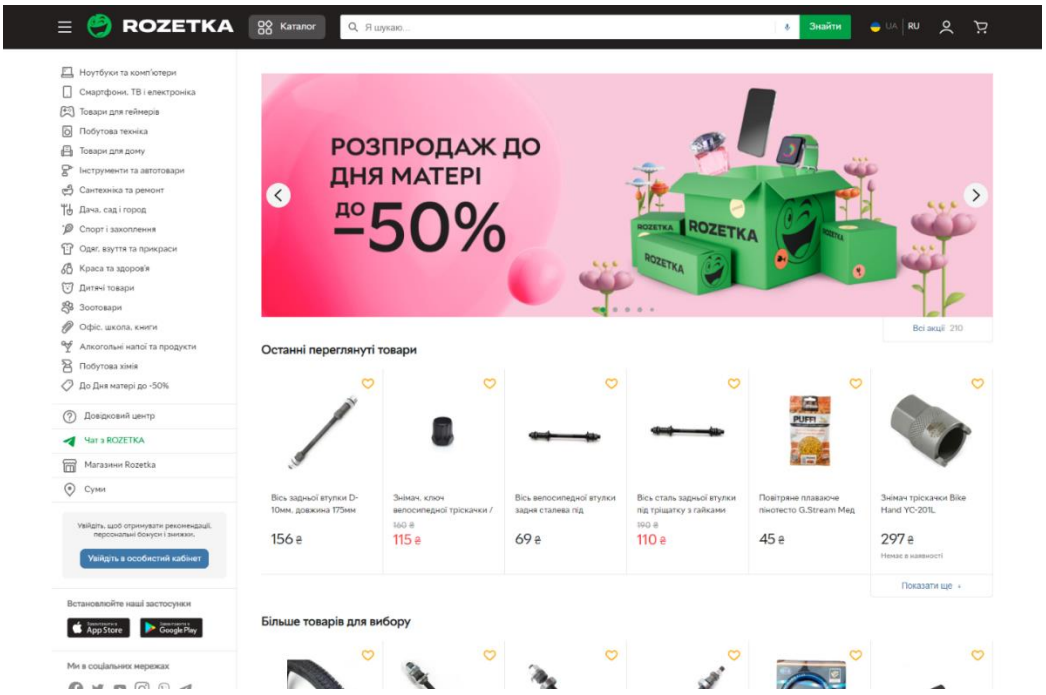


Рисунок 1.1 – Інтернет-магазин Rozetka

Yes, original – це не просто інтернет-магазин, а велика мультибрендова платформа, яка пропонує оригінальне взуття, одяг та аксесуари від всесвітньо відомих виробників. Магазин гарантує високу якість і оригінальність своєї продукції. Серед переваг цього сайту можна відзначити привабливий дизайн,

наявність оригінальних товарів, якісні фотографії і швидку доставку (рис. 1.2). Однак серед недоліків можна відзначити часту відсутність товару на складі[6].

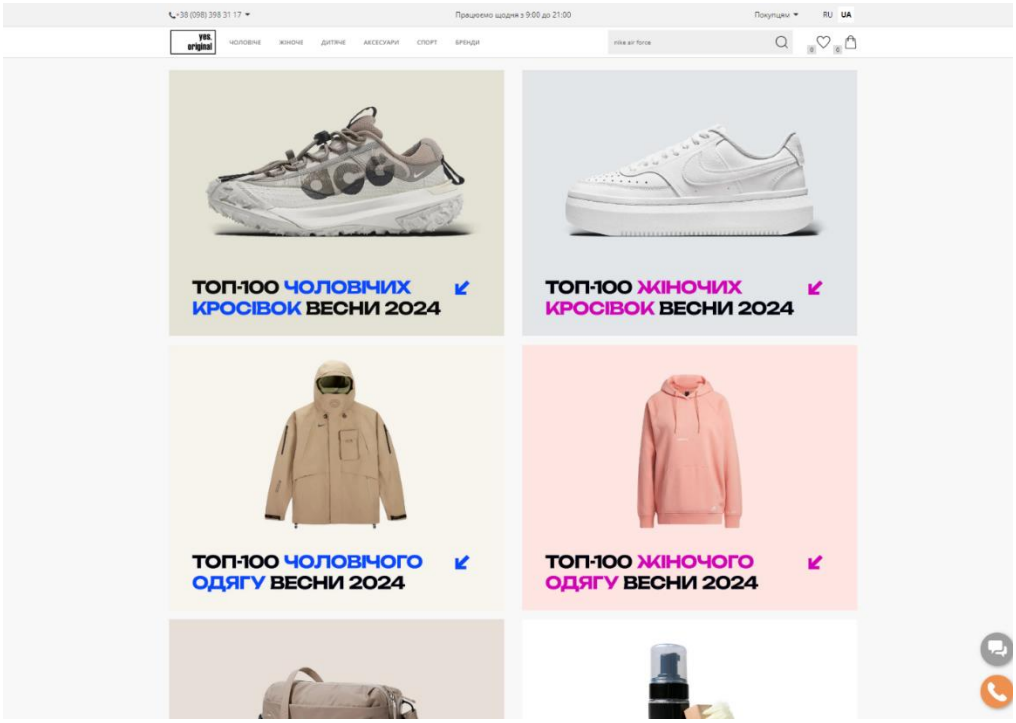


Рисунок 1.2 – Інтернет-магазин yes, original

Аналіз конкурентів є важливим етапом для розвитку інтернет-магазину, оскільки допомагає зрозуміти конкурентну обстановку та розробити стратегію, спрямовану на привертання та збереження клієнтів.

1.3 Постановка задачі

Метою роботи: є розробка та програмна реалізація інтернет-магазину одягу та взуття. Для досягнення поставленої мети необхідно буде виконати наступні задачі:

1. Аналіз та огляд популярних веб-магазинів (провести дослідження ринку продажу взуття та одягу, а також зробити аналіз враховуючи плюси та мінуси переглянутих інтернет-магазинів).

2. Аналіз та підбір методів для вирішення задач (проаналізувати методи вирішення задач та вибрати оптимальні).

3. Проектування інтернет-магазину (створення прототипів сторінок сайту, Use Case діаграми тощо).

4. Розробка frontend частини (розроблений дизайн має слідувати сучасним стандартам та бути зручним для користувачів).

5. Розробка backend частини (реалізація необхідного функціоналу: каталог продукції, кошик, особистий кабінет, оформлення замовлення тощо).

6. Тестування інтернет-магазину (провести тестування функціоналу на коректність).

Ці задачі допоможуть у розробці та впроваджені інтернет-магазина з метою його успішного функціонування.

2. ВИБІР МЕТОДУ РОЗВ’ЯЗАННЯ ЗАДАЧІ

2.1 Python. Що таке Python фреймворки, які фреймворки існують та чому саме Django

Python - це одна з найпотужніших та популярніших мов програмування, широко використовувана розробниками для створення застосунків та сайтів. Одна з основних переваг Python - існування потужної екосистеми фреймворків, яка надає програмістам актуальні засоби і рішення прискорюючи процес розробки.

Фреймворки Python забезпечують розробників наборами інструментів, бібліотек та шаблонів, полегшуючи створення застосунків, веб-сайтів, а також інших програмних альтернатив, що забезпечують структуру та архітектуру створення додатків, а також підтримку маршрутизації URL-адрес, управління базами даних, обробку форм, аутентифікація користувача тощо[7].

Python фреймворки та їх переваги:

- Django – full-stack фреймворк який має доступний відкритий код. Зарекомендував себе як один з головних і популярних серед розробників які використовують мову Python. За досить короткочасний період сприяє переходу від прототипу до готового вирішення задач, адже основа роль яку він виконує – завдяки бібліотекам та об'єднанням автоматизує процеси та прискорює роботу. Здебільшого рішення мають коробковий вигляд, їх треба лише використовувати, а не вигадувати щось нове.
- CherryPy – це мікро-фреймворк. Його призначення - вирішення конкретних задач, має унікальну здатність запускати додатки на будь-якій ОС.
- Flask – це мікрофреймворк, який за допомогою Python дозволяє створювати прості проекти даючи можливість масштабути їх до складних програм. Під поняттям "мікрофреймворк" можна трактувати можливість встановлення обраних бібліотек та інструментів, які

потрібні для виконання певного завдання, адже одразу в комплекті їх немає.

- Web2Py – об'ємний фреймворк Python, що має свій IDEwhich, який містить: редактор коду, debugger, deploy. Не потребує якихось налаштувань або встановлень, працюючи при цьому відмінно, надає можливість роботи на різних платформах. Про рівень безпеки даних хвилюватися не потрібно, бо тут все на вищому рівні.
- Pyramid – це Python фреймворк, який застосовується для розв'язання багатофункціональних задач, а також розробки досить складних об'єктів. В основному він використовується професійними розробниками, застосовують його для маршутизації та ідентифікації[7].

Ці фреймворки входять в топ найпопулярніших, що підтверджується графіком (рис. 2.1)

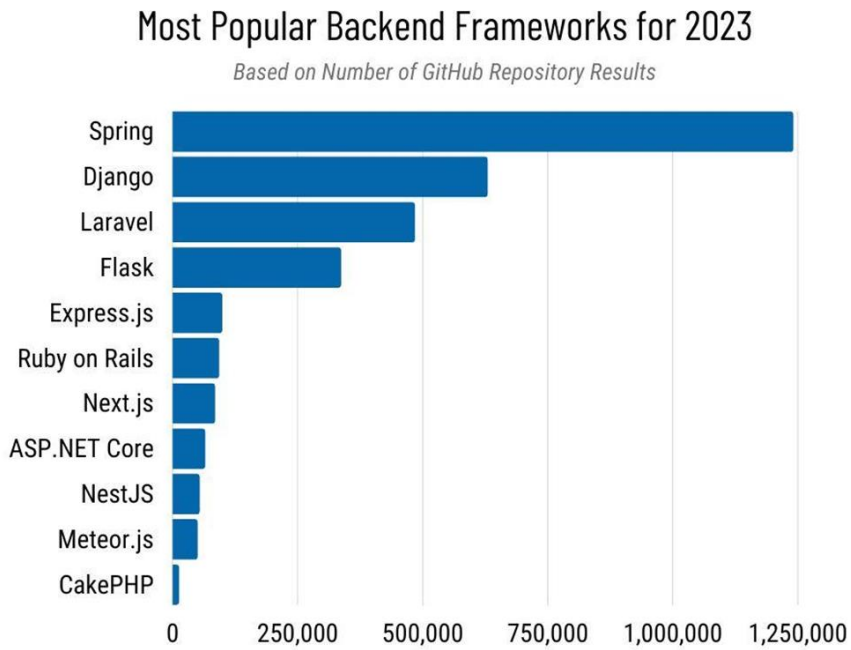


Рисунок 2.1 – Графік найбільш популярних фреймворків[9]

Головні переваги Django включають:

1. Універсальність:
   - Використання для створення різних веб-додатків:
     - Сайти електронної комерції;
     - Соціальні мережі;
     - Веб-застосунки з інтерактивними функціями.
2. Безпека:
   - Розроблений з урахуванням безпеки;
   - Захист від поширених веб-загроз (ін'єкції SQL, CSRF (міжсайтова підробка запитів) та XSS (міжсайтовий скриптінг), які допомагають захистити додатки від популярних видів атак.).
3. Швидкість розробки:
   - Django пропонує структурований підхід до розробки, який економить час;
   - Дозволяє швидко створювати прототипи та випускати продукти;
   - Вбудовані функціональні можливості для маршрутизації URL, шаблонів, обробки форм та адміністрування.
4. Документація та спільнота:
   - Чудова документація;
   - Активна спільнота розробників для допомоги у вирішенні проблем.
5. Масштабування:
   - Підходить для проектів будь-якої складності;
   - Структура та функції дозволяють легко масштабувати додаток.
6. Багата екосистема:
   - Економить час на написанні коду з нуля;
   - Велика та активна спільнота;
   - Численні пакети та бібліотеки для розширення функціоналу[8].

2.2 Налаштування середовища

Головна мета Django полягає в полегшенні процесу розробки веб-додатків шляхом надання готових інструментів та шаблонів, які дозволяють розробникам швидко створювати функціональні веб-додатки з великою кількістю функціональних можливостей[9].

Для створення проекту Django у середовищі PyCharm слід виконати наступні кроки:

1. Встановити Python: Якщо Python 3.10 ще не встановлено на вашому комп'ютері, встановіть його.
2. Встановити PyCharm: Якщо ви ще не маєте PyCharm, встановіть її.
3. Відкрити PyCharm: Запустити PyCharm на вашому комп'ютері.
4. У вікні запуску обрати "New Project" (рис. 2.2).

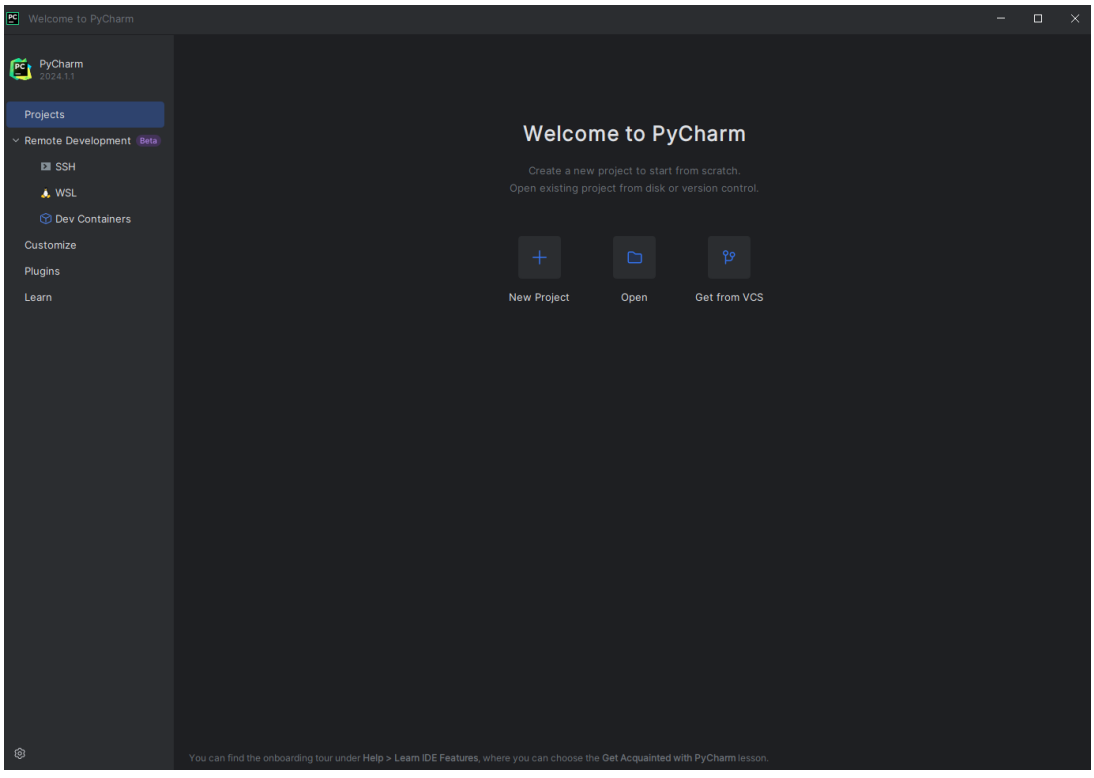


Рисунок 2.2 – Вікно програми PyCharm (створення нового проекту)

5. Обираємо фреймворк Django та вводимо назву для вашого проекту. Для створення проекту натискаємо "Create" (рис. 2.3).

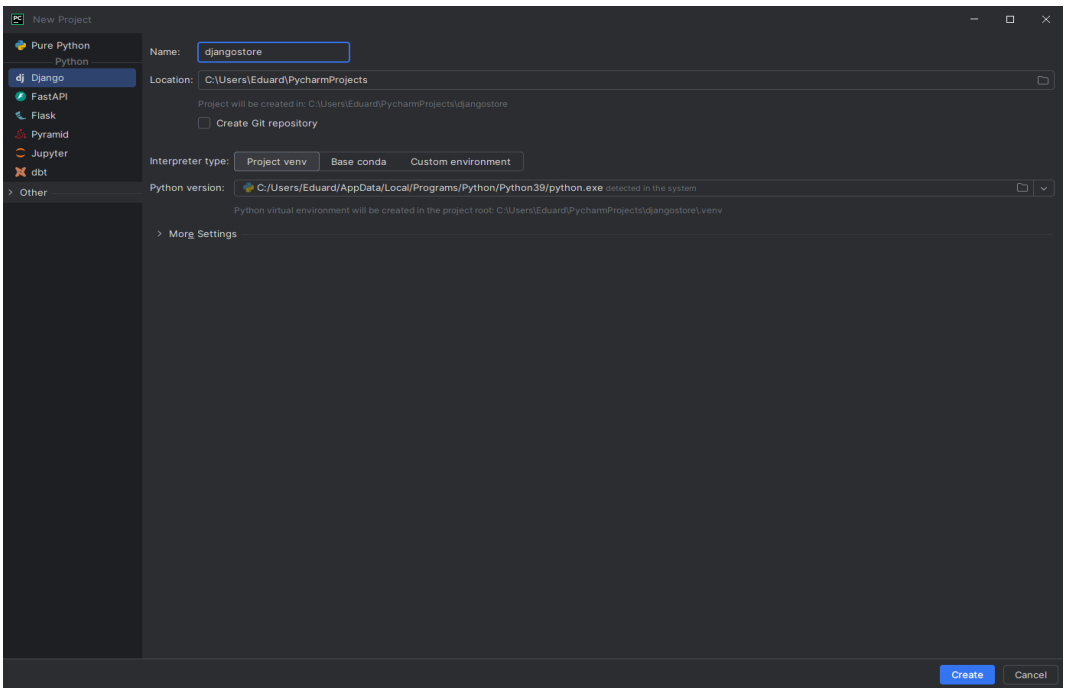


Рисунок 2.3 – Створення проекту

6. Маємо отримати результат – встановлений проект. (рис. 2.4)

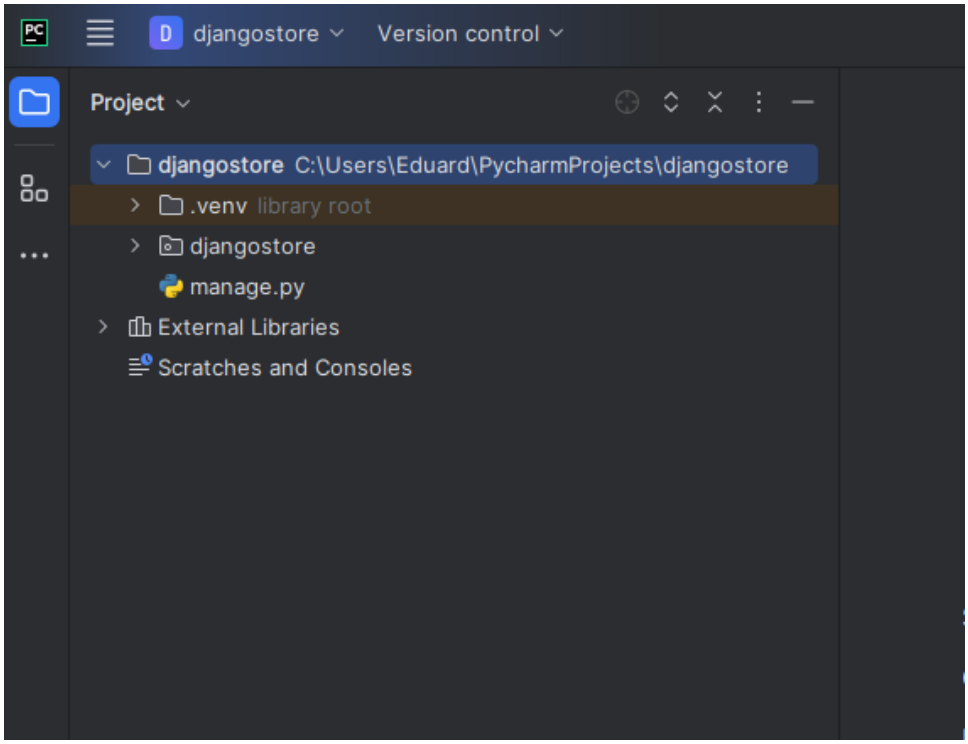


Рисунок 2.4 – Створений проект

7. Перевіряємо встановлену версію Django командою Django.VERSION та версію Python.

8. Запуск проекту: Щоб запустити ваш проект Django, використовуйте команду python manage.py runserver. (рис. 2.5)

Рисунок 2.5 – Запуск проекту

2.3 Порівняння баз даних

Порівняємо PostgreSQL, MongoDB та MySQL за такими критеріями:

Масштабованість:

- PostgreSQL: Підтримує розподілені бази даних та можливості реплікації, що дозволяє масштабувати систему з ростом обсягу даних та навантаження.
- MongoDB: Легко масштабується горизонтально за рахунок розподіленої архітектури та шардингу.
- MySQL: Підтримує реплікацію та горизонтальний шардування, але не так ефективно як MongoDB або PostgreSQL.

Використання мови запитів:

- PostgreSQL та MySQL: Використовують мову запитів SQL.
- MongoDB: Використовує декларативну мову запитів, яка називається MongoDB Query Language (MQL).

Підтримка транзакцій:

- PostgreSQL та MySQL: Підтримують транзакції з комітами та відкатами.

- MongoDB: Підтримує транзакції з початками та збереженнями, але не підтримує транзакції між декількома колекціями у різних базах даних.

Схема даних:

- PostgreSQL: Використовує фіксовану схему даних зі строгими обмеженнями на типи даних та зв'язки між таблицями.
- MongoDB: Не має жорсткої схеми даних, дозволяючи зберігати різні типи даних в одному документі.
- MySQL: Використовує фіксовану схему даних, але може бути менш жорсткою, ніж PostgreSQL, з можливістю зберігання некоректних даних у випадку використання деяких типів таблиць.

Загалом, кожна з цих систем має свої особливості та відповідає на різні вимоги проектів. PostgreSQL і MySQL зазвичай краще підходять для традиційних реляційних схем даних, тоді як MongoDB може бути кращим вибором для проектів з потребою у гнучкості схеми та швидкості розробки[11].

2.4 Вибір бази даних

PostgreSQL – це потужна об'єктно-реляційна система управління базами даних (СКБД), яка використовується для зберігання та обробки структурованої інформації.

Переваги використання PostgreSQL для інтернет-магазинів:

- Гнучкість та розширюваність: має широкий спектр функцій та можливостей, які дозволяють адаптувати базу даних до потреб вашого інтернет-магазину. Він також підтримує багато розширень, які дозволяють розширювати його функціональність за потреби.
- Масштабованість: підтримує масштабування, що означає, що ви можете почати з невеликого інтернет-магазину і поступово збільшувати його розмір та функціональність з ростом вашого бізнесу.

- Підтримка широкого спектру типів даних: підтримує різні типи даних, включаючи текст, числа, JSON, географічні дані та багато іншого. Це дозволяє зберігати та обробляти різноманітну інформацію в вашому інтернет-магазині.
- Безпека: PostgreSQL має вбудовані механізми безпеки, що дозволяють захистити ваші дані від несанкціонованого доступу, втрати чи порушення цілісності.
- Надійність і стабільність: PostgreSQL відомий своєю надійністю та стабільністю. Це означає, що ваш інтернет-магазин буде працювати безперебійно, навіть при високих навантаженнях та при великій кількості користувачів [10].

Загалом, обираючи PostgreSQL для свого інтернет-магазину, ми отримуємо надійну, масштабовану та гнучку базу даних, яка забезпечить ефективну роботу нашого бізнесу та захистить наші дані.

3 ІНФОРМАЦІЙНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Проектування інтернет-магазину

Проектування інтернет-магазину – необхідний крок, бо це основа на якій буде створюватись сайт.

1) Прототип веб-сайту – це перша модель або концепція веб-сайту, яка відображає його основну структуру та функціональність перед активною розробкою. Прототип допомагає візуалізувати ідеї та концепції, що стоять за вашим веб-сайтом, дозволяючи команді проекту та зацікавленим сторонам краще розуміти його вигляд та функціональність на ранніх етапах розробки.

Основні переваги прототипів включають:

- Візуалізацію концепції: Прототип допомагає візуально представити ідеї та концепції, які визначають основу вашого веб-сайту, допомагаючи команді проекту та клієнту краще розібратися, як виглядатиме та працюватиме сайт на ранніх стадіях розробки.
- Збір фідбеку: Прототип можна представити клієнту, стейкхолдерам та користувачам для збору фідбеку. Це дозволяє виправити можливі проблеми та недоліки ще до початку активної розробки, що зменшує витрати на подальше виправлення помилок.
- Визначення функціональності: Прототип допомагає визначити, які функції та можливості повинні бути на сайті. Це важливо для уточнення вимог та визначення обсягу робіт.
- Оптимізація інтерфейсу: За допомогою прототипу ви можете експериментувати з розташуванням елементів на сторінці, вибором кольорів, шрифтів та інших дизайнерських аспектів для досягнення найкращого користувацького досвіду.
- Подальший розвиток: Прототип може бути використаний як основа для подальшої розробки сайту, включаючи створення HTML/CSS-версії або інтеграцію з веб-фреймворками.

- Комунікація з зацікавленими сторонами: Прототип є ефективним інструментом комунікації з командою проекту, клієнтами та іншими учасниками.

Прототипи можуть бути створені різними способами, включаючи паперові макети, друковані зразки, веб-макети або застосування спеціального програмного забезпечення для створення прототипів. Вони можуть бути простими або деталізованими, залежно від потреб проекту. У будь-якому випадку створення прототипу сприяє покращенню якості та успіху веб-проекту[12].

Орієнтуючись на технічне завдання було розроблено прототипи сторінок сайту, а саме:

Прототипи головної сторінки (рис. 3.1), каталог (рис. 3.2), про нас (рис. 3.3), особистий кабінет (рис. 3.4), сторінка товару (рис. 3.5), кошик (рис. 3.6), оформлення замовлення (рис. 3.7).

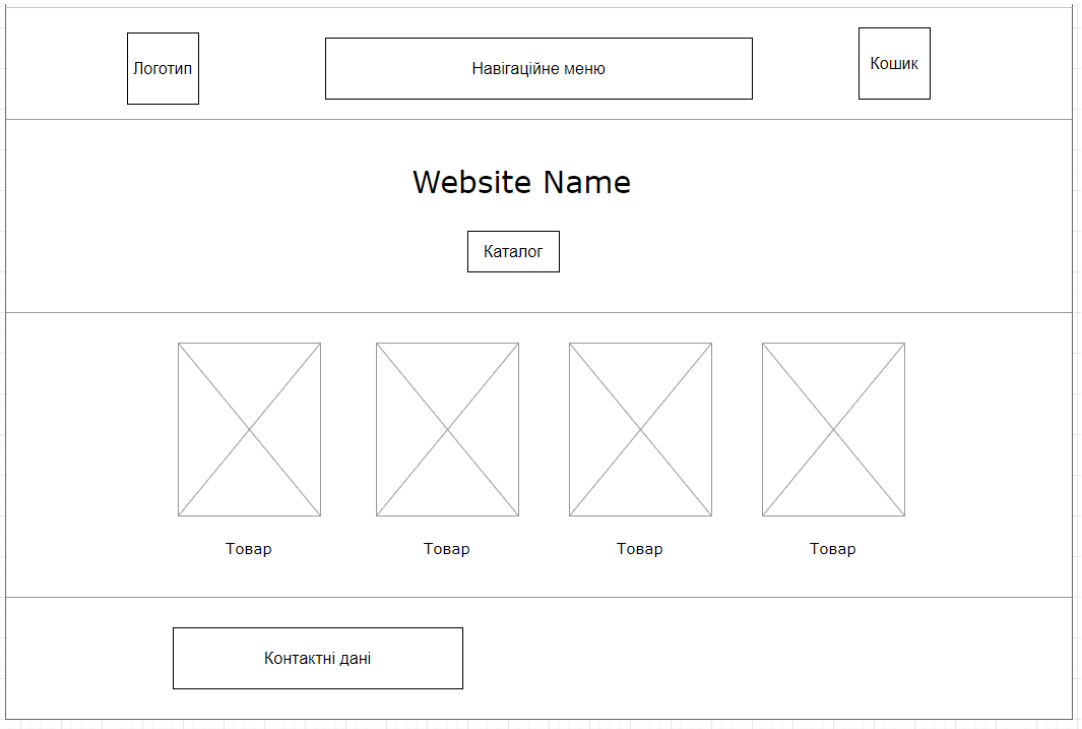


Рисунок 3.1 – Прототип головної сторінки сайту

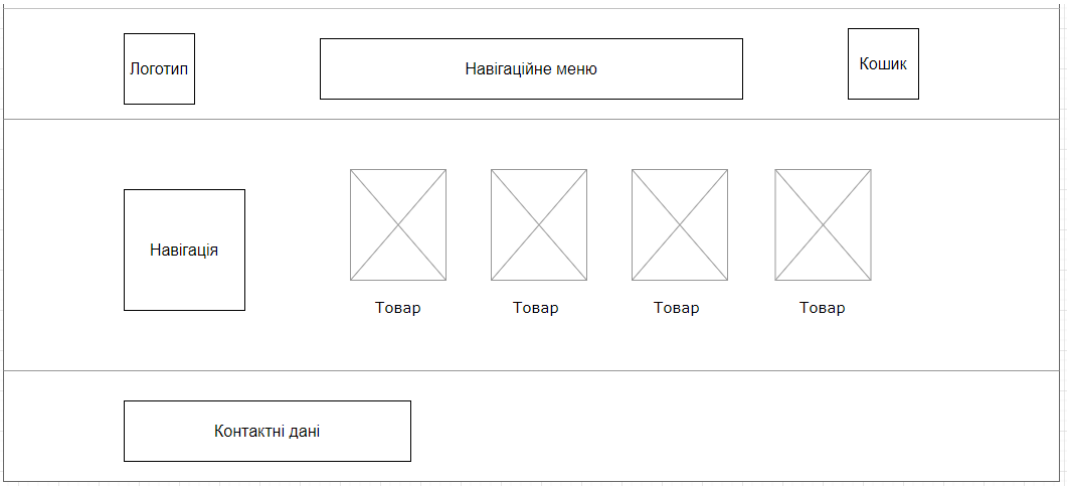


Рисунок 3.2 – Прототип каталогу

Рисунок 3.3 – Прототип сторінки "Про нас"

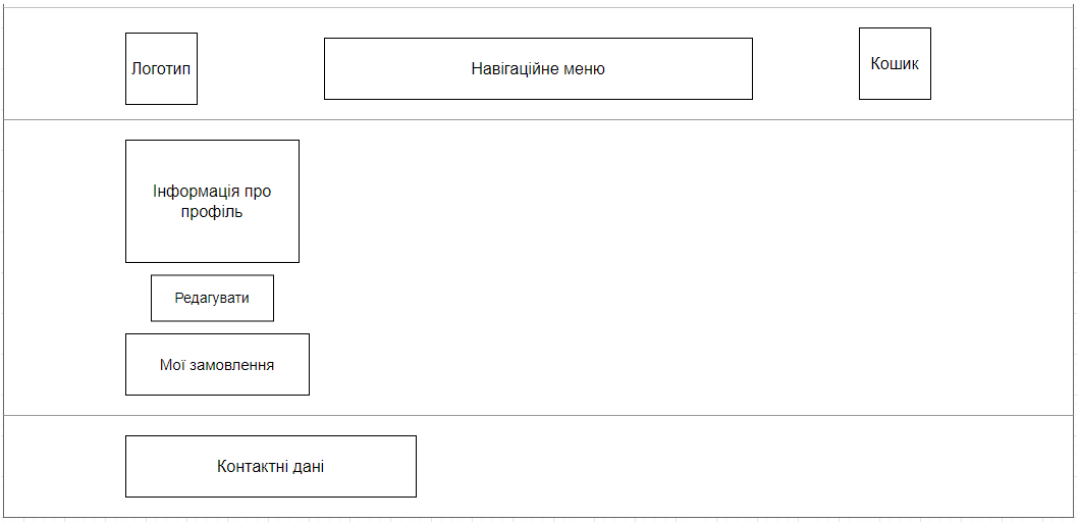


Рисунок 3.4 – Прототип особистого кабінету

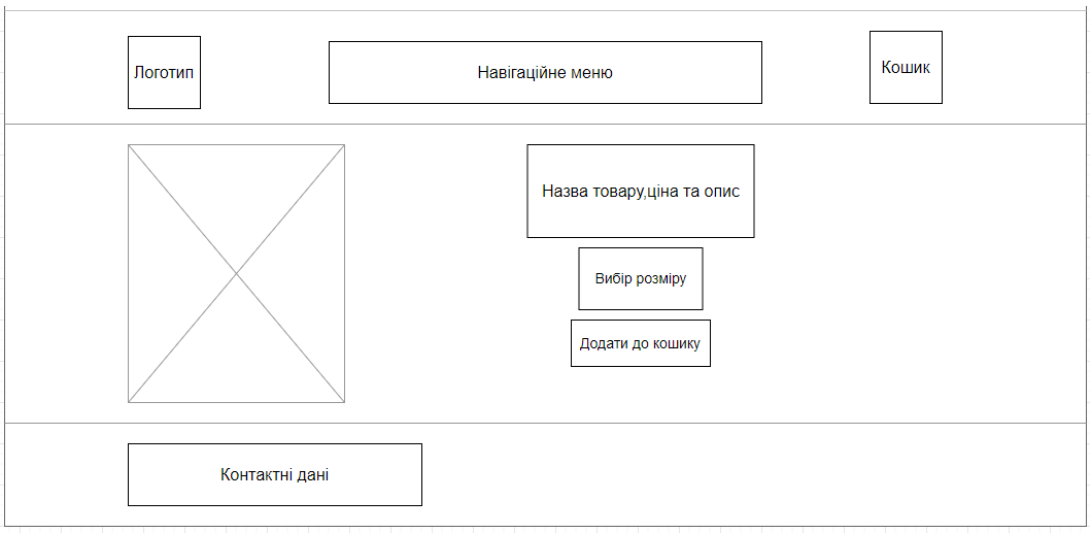


Рисунок 3.5 – Прототип сторінки товару

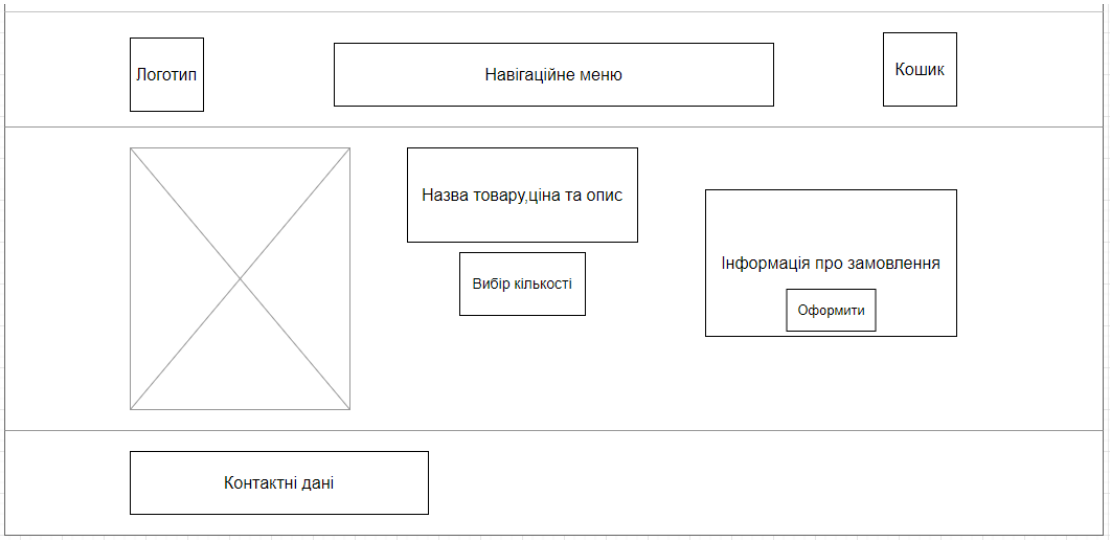
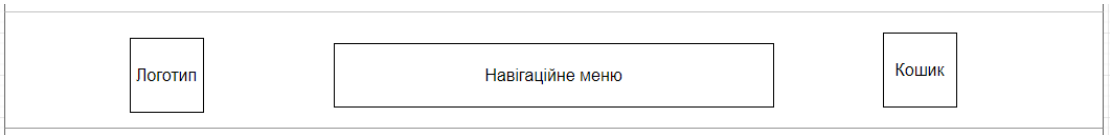


Рисунок 3.6 – Прототип кошика



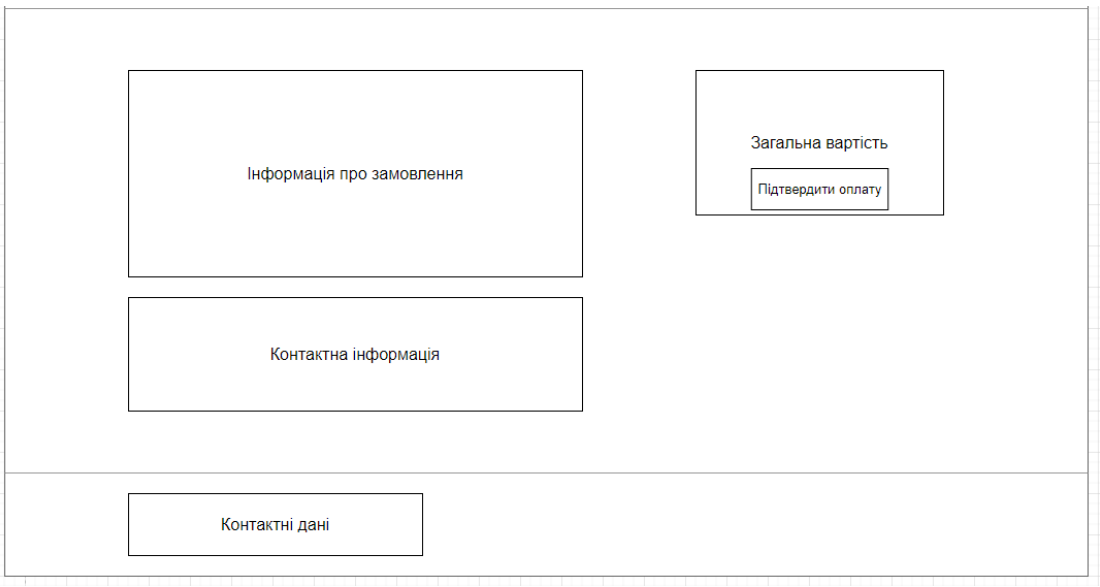


Рисунок 3.7 – Прототип сторінки оформлення замовлень

2) UML (Unified Modeling Language) діаграма – це графічне зображення, яке використовується для візуалізації, проектування та аналізу програмних систем. Вона містить різні типи діаграм, такі як діаграми класів, взаємодії, послідовності, станів та інші, які допомагають розуміти структуру та функціональність системи[13].

UML діаграма класів нашого інтернет-магазину (рис. 3.8).

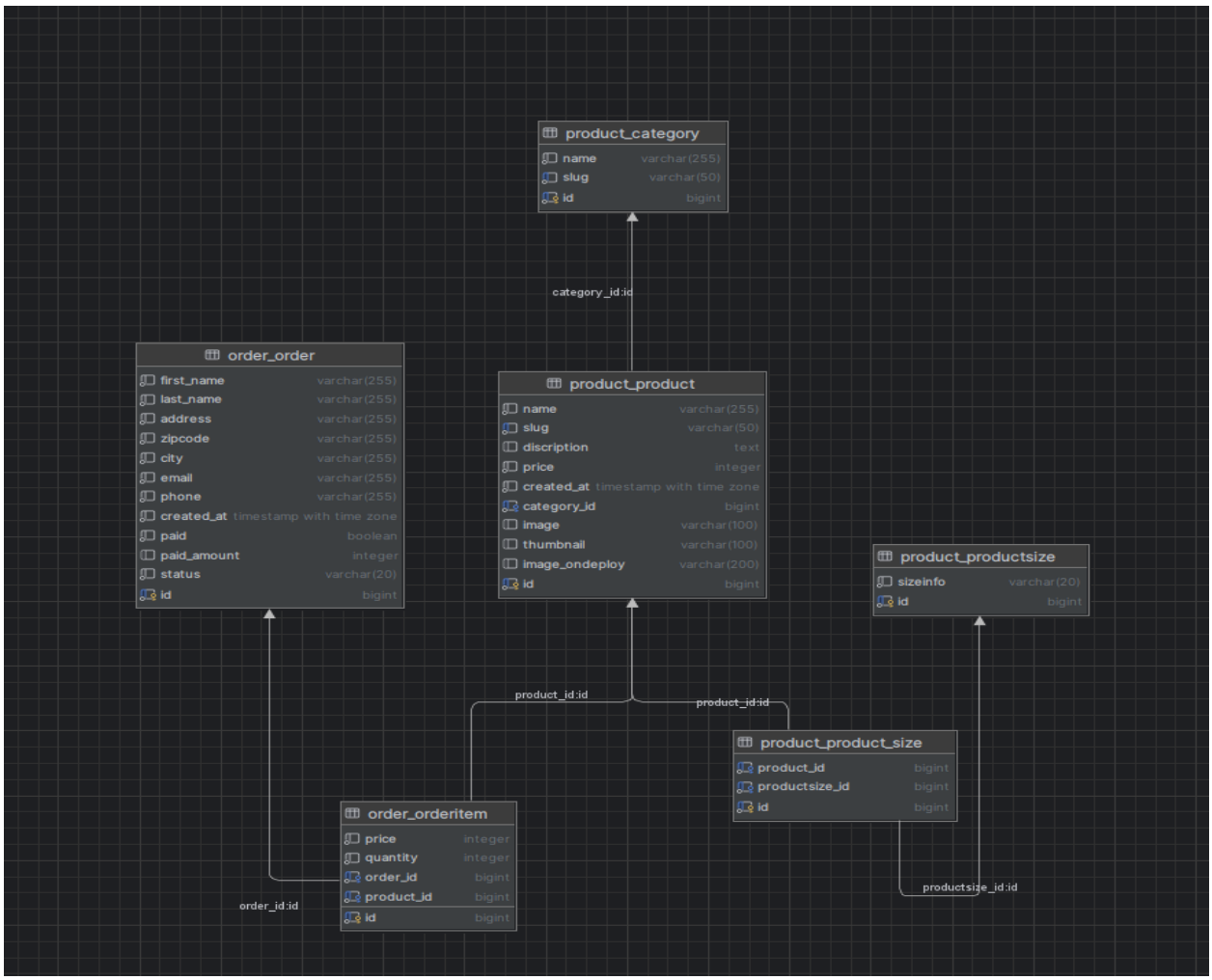


Рисунок 3.8 – UML діаграма класів

3) Діаграма варіантів використання (Use Case Diagram) в UML – це графічне представлення функціональних можливостей системи, тобто того, що вона може робити для своїх користувачів.

Вона показує:

- Акторів: зовнішні сутності, які взаємодіють із системою, наприклад, користувачів, інші системи або пристрої.
- Варіанти використання: основні функціональні можливості системи, які описують, як актори досягають своїх цілей за допомогою взаємодії з системою.
- Взаємозв'язки: між акторами та варіантами використання.

На діаграмі варіантів використання актори зображуються у вигляді фігурок людей, варіанти використання - овалами, а зв'язки - суцільними лініями[14].

Розроблено дві Use Case діаграми:

- Для клієнта (рис. 3.9).
- Для адміністратора (рис. 3.10).

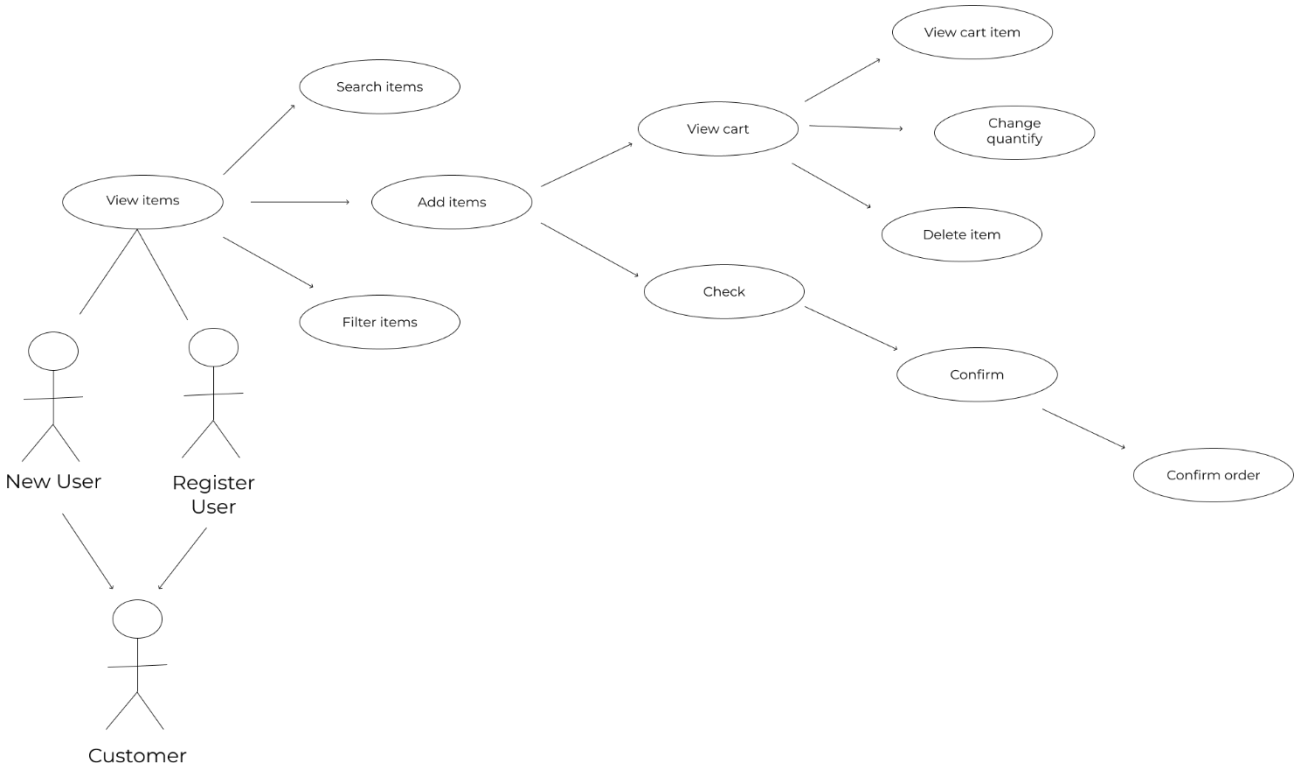


Рисунок 3.9 – Use Case діаграма для клієнта

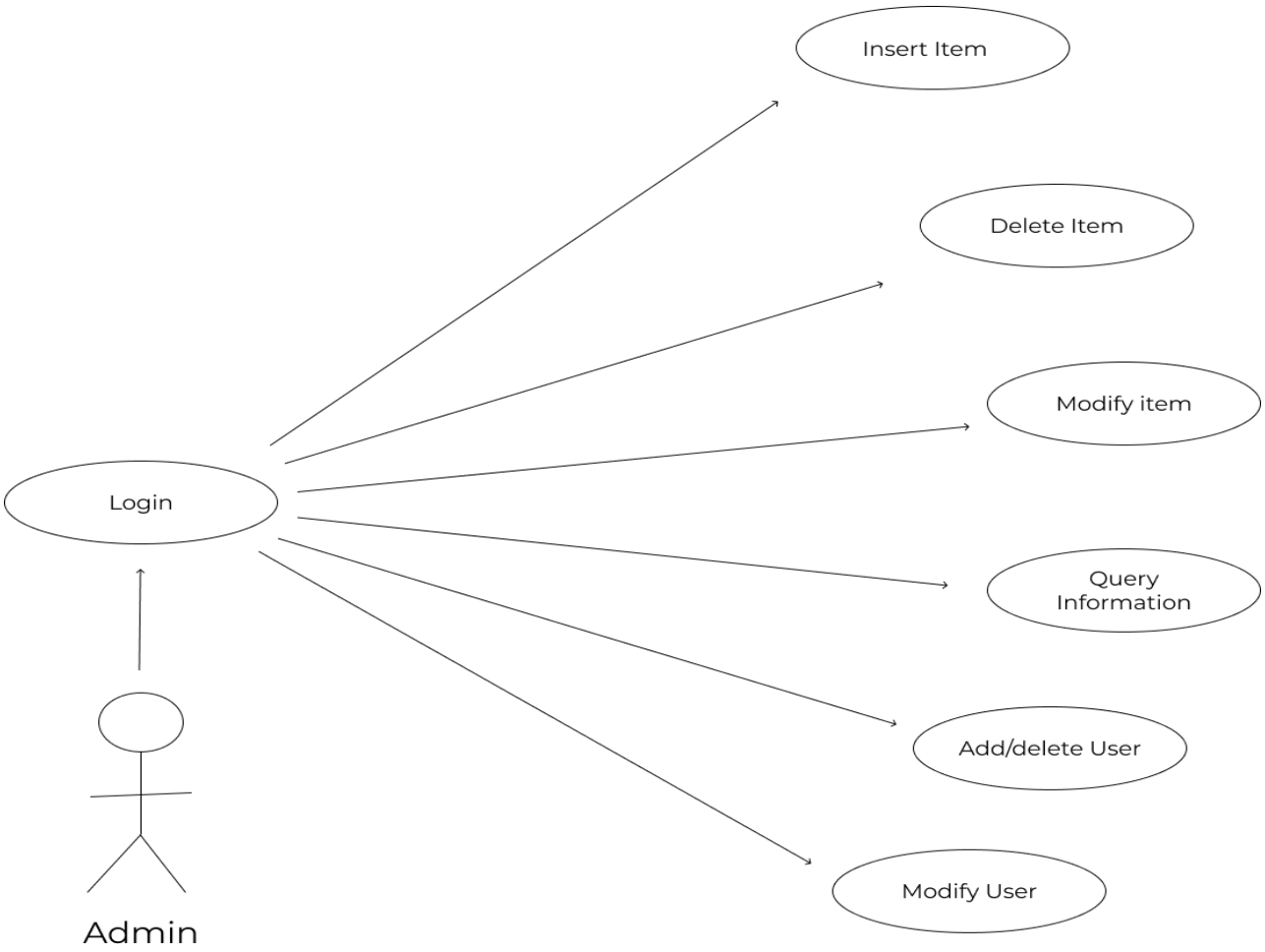


Рисунок 3.10 – Use Case діаграма для адміністратора

3.2 Проектування бази даних

Проектування бази даних - це процес створення структури та організації бази даних з метою ефективного зберігання та управління даними. Основні кроки проектування включають визначення потреб користувачів, аналіз вимог до даних, створення схеми бази даних (за допомогою концептуальних та логічних моделей даних), нормалізацію, що спрямована на зменшення дублювання даних та забезпечення їх консистентності, і, нарешті, реалізацію фізичної структури бази даних у вибраній СУБД. Важливо враховувати вимоги до продуктивності, безпеки та масштабованості при проектуванні бази даних.

ER-діаграма (Entity-Relationship Diagram), також відома як діаграма зв'язків сутностей, - це графічний спосіб представлення структури бази даних.

Вона показує:

- Сутності: основні об'єкти, з якими працює система, наприклад, клієнти, замовлення або товари.
- Атрибути: характеристики, які описують кожну сутність.
- Зв'язки: стосунки між сутностями.

На ER-діаграмі сутності зображуються прямокутниками, атрибути - списком всередині прямокутників, а зв'язки - лініями між прямокутниками[15].

Спроектована ER-діаграма для нашого інтернет-магазину(рис. 3.11).

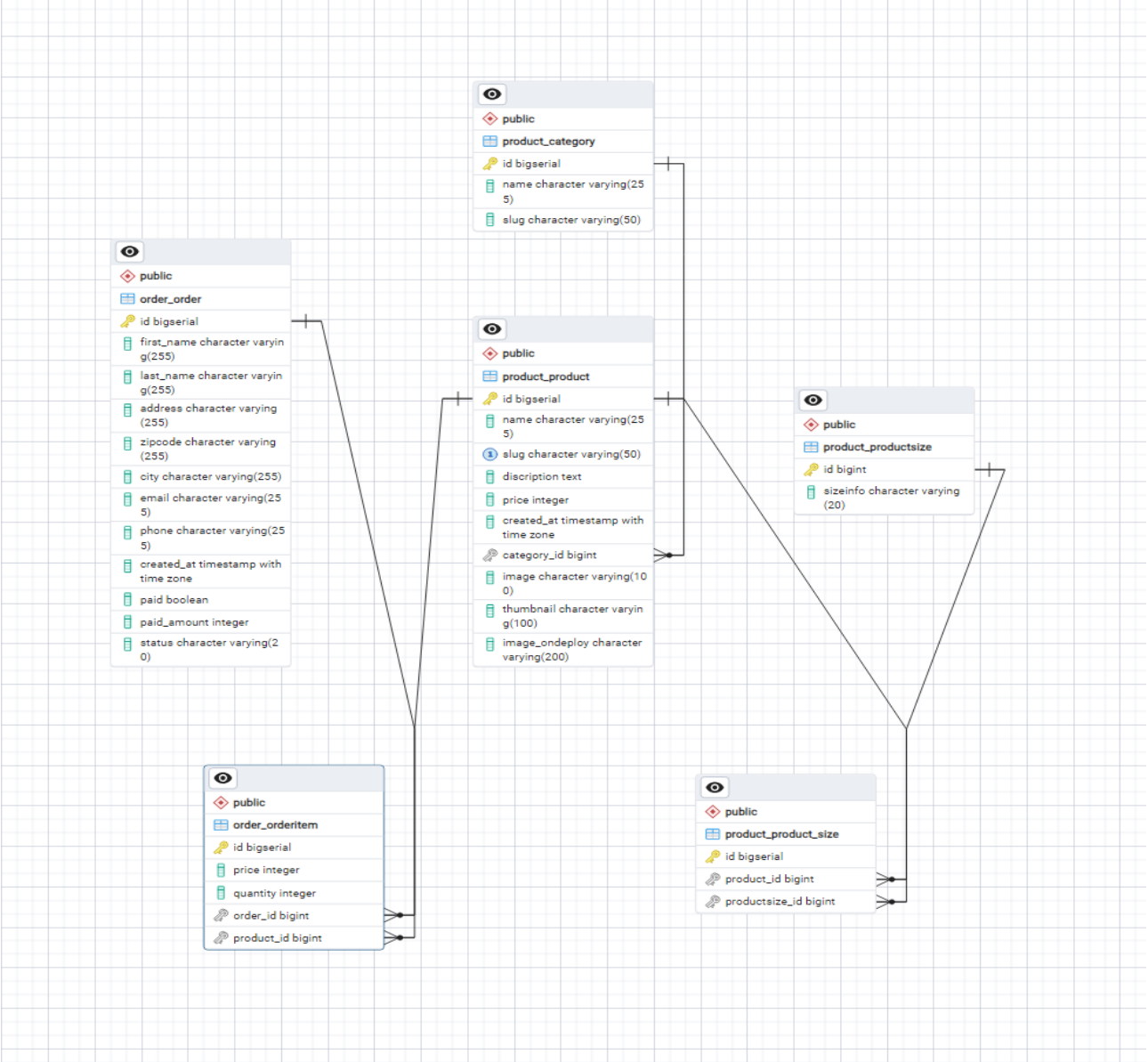


Рисунок 3.11 – ER-діаграма

Після успішного проектування прототипу сайту, UML та Use Case діаграм та бази даних - переходимо до реалізації сайту.

3.3 Реалізація сайту

Перелік створених сторінок при написанні сайту:

- Головна сторінка;
- Каталог;
- Про нас;
- Особистий кабінет;
- Сторінка товару;
- Кошик;
- Оформлення замовлення;
- Форми реєстрації та авторизації.

Клієнтська частина сайту:

При відкритті сайту відображається головна сторінка (рис. 3.12), на якій містяться всі товари у наявності. У верхній частині зліва вкладки "Магазин" та "Про нас", а справа розміщений кошик, який показує кількість обраних товарів. У нижній частині сторінки розміщена інформація про адресу магазину та контактні дані.

Повний код додатку core/views.py знаходиться в "Додаток А".

```
def frontpage(request):
    products = Product.objects.all()[0:8]
    return render(request, 'core/frontpage.html', {'products': products})
```

frontpage(request) – ця функція обробляє головну сторінку сайту. Вона отримує останні 8 продуктів з бази даних, відрендерює шаблон core/frontpage.html(Додаток Б) і передає список продуктів в контекст.

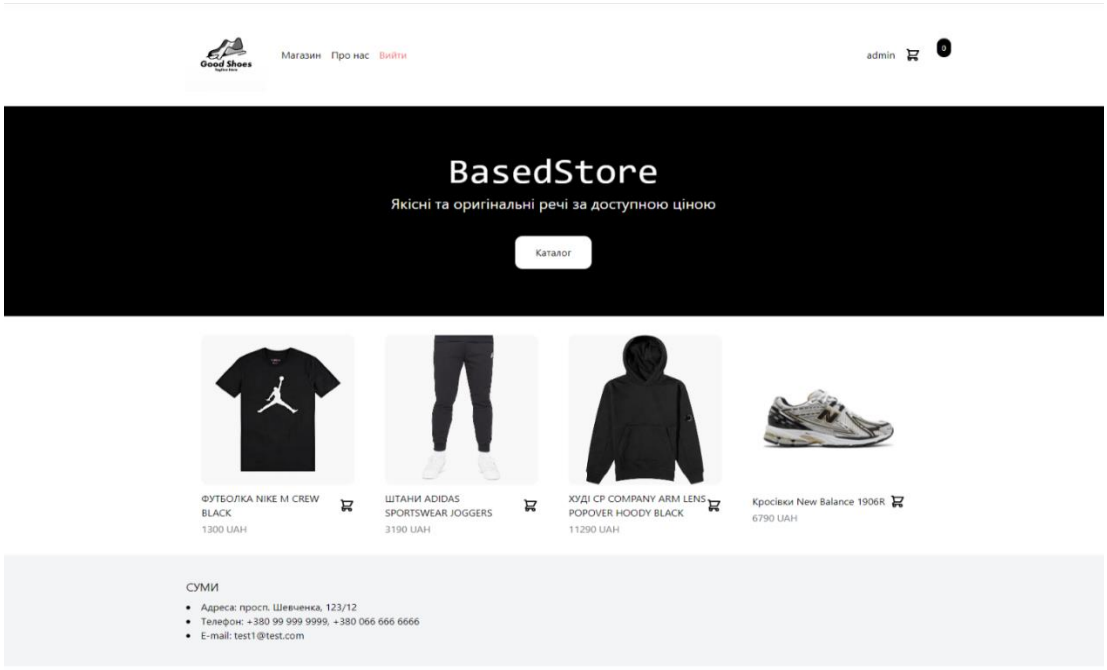


Рисунок 3.12 – Головна сторінка

Сторінка каталог (рис. 3.13) містить обрані товари відповідно до категорії, а також присутня навігація (рис. 3.14) для зручного пошуку товару за назвою.

```
def shop(request):
    categories = Category.objects.all()
    products = Product.objects.all(
    active_category = request.GET.get('category', '')
    if active_category:
        products = products.filter(category__slug=active_category)
    query = request.GET.get('query', '')
    if query:
        products = products.filter(Q(name__icontains=query) | Q(discription__icontains=query))
    context = {
        'categories': categories,
        'products': products,
        'active_category': active_category,
    }
    return render(request, 'core/shop.html', context)
```

shop(request) – ця функція обробляє сторінку магазину. Отримує всі категорії та продукти з бази даних, активну категорію з GET-параметрів (якщо active_category не пуста, фільтрує продукти за категорією), отримує рядок пошуку з GET-параметрів (якщо query не пуста, фільтрує продукти за назвою або описом). Створює контекстний словник з отриманими даними: категорії, продукти, активна категорія. Відрендерює шаблон core/shop.html(Додаток Б).

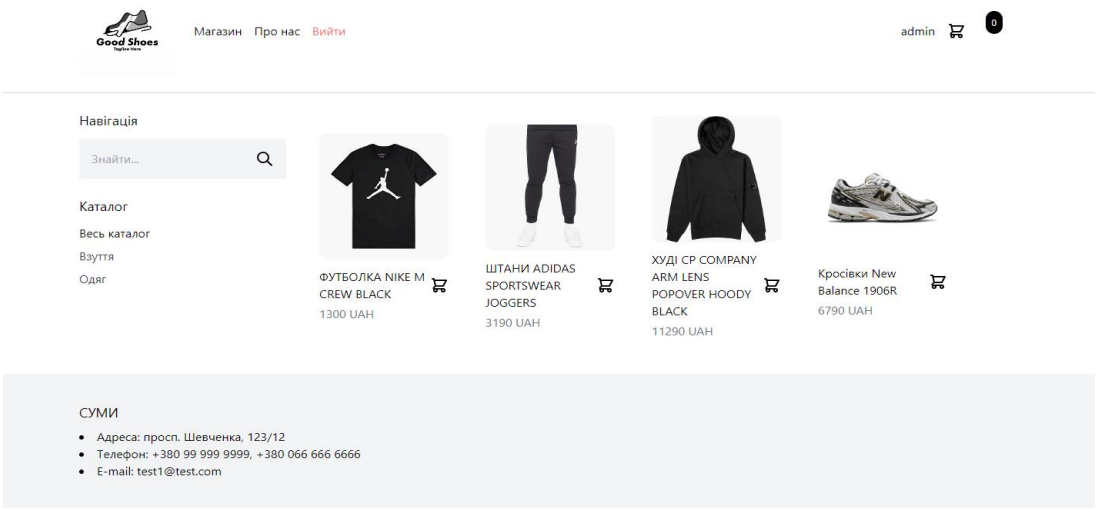


Рисунок 3.13 – Каталог

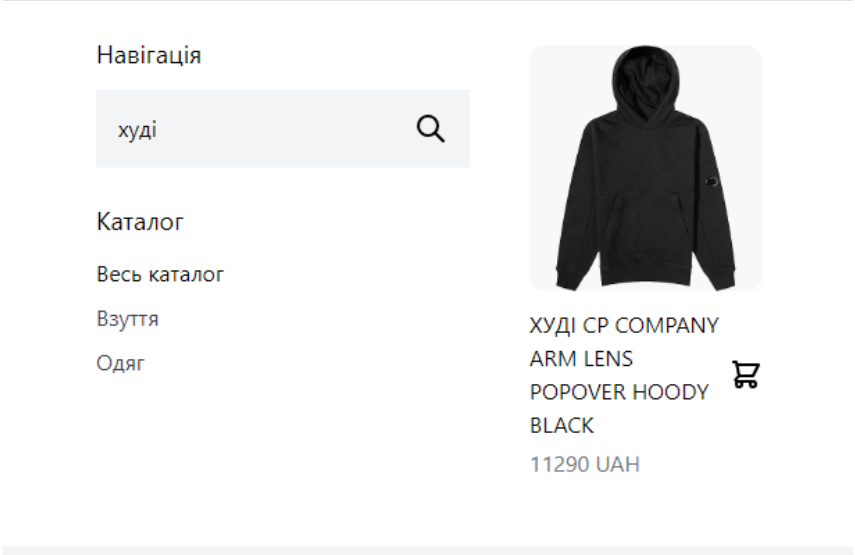


Рисунок 3.14 – Навігація

Сторінка про нас (рис. 3.15) містить інформацію про магазин та переваги серед конкурентів.

```
def aboutus(request):
    return render(request, 'core/aboutus.html')
```

aboutus(request) – ця функція обробляє сторінку "Про нас". Відрендерює шаблон core/aboutus.html(Додаток Б).

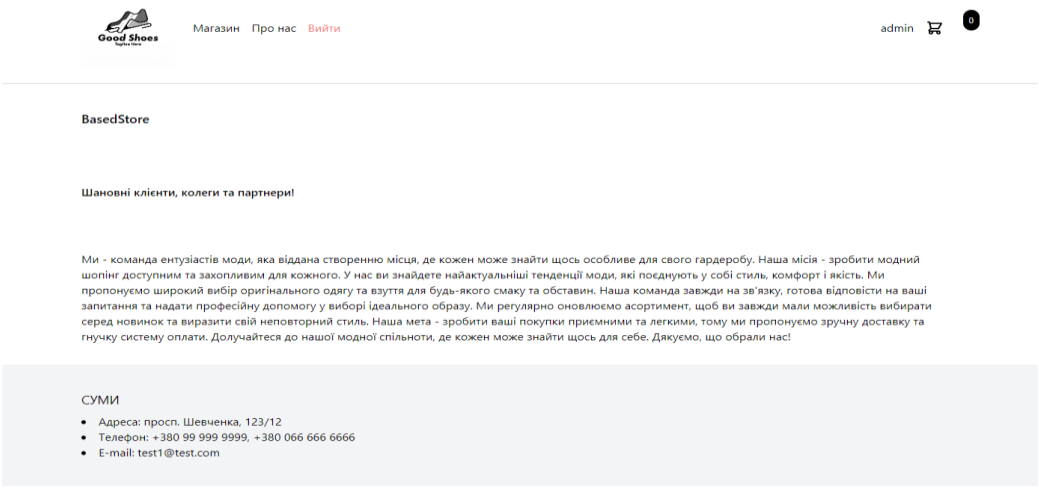


Рисунок 3.15 – Про нас

"Особистий кабінет" (рис. 3.16) містить інформацію про користувача: логін, пошта, ім’я та прізвище, а також замовлення які були зроблені за весь час. Присутня можливість редагувати профіль.

```
@login_required
def myaccount(request):
    return render(request, 'core/myaccount.html')
@login_required
def edit_myaccount(request):
    """Редагування профіля"""
    if request.method == 'POST':
        user = request.user
        user.first_name = request.POST.get('first_name')
        user.last_name = request.POST.get('last_name')
        user.username = request.POST.get('username')
```

```
user.email = request.POST.get('email')
        user.save()
        return redirect('myaccount')
    return render(request, 'core/edit_myaccount.html')
```

@myaccount(request) – функція обробляє сторінку "Мій профіль". Декоратор @login_required перевіряє чи користувач авторизований, перш ніж дозволити йому перейти на сторінку профілю. Відрендерює шаблон core/myaccount.html(Додаток Б).

@edit_myaccount(request) – функція обробляє редагування профілю користувача. Декоратор login_required перевіряє чи користувач авторизований, перш ніж дозволити йому перейти на сторінку профілю. Перевіряє метод HTTP-запиту (request.method). Відрендерює шаблон core/edit_myaccount.html (Додаток Б).

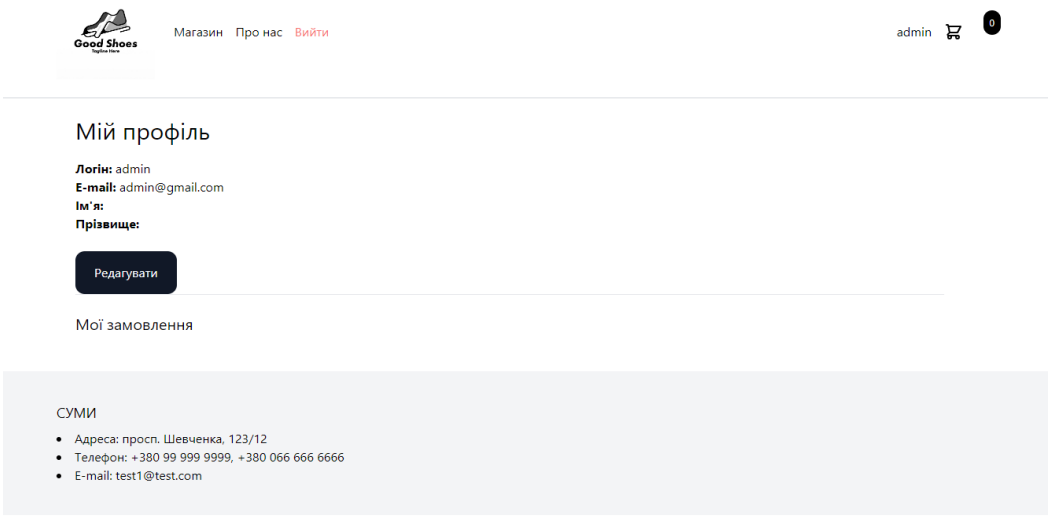


Рисунок 3.16 – Особистий кабінет

Кожен товар можна відкрити на новій сторінці для зручного перегляду. Сторінка товару (рис. 3.17) містить детальну інформацію про нього: назва, фото, ціна, опис, розмір. Присутня можливість додавання до кошику.

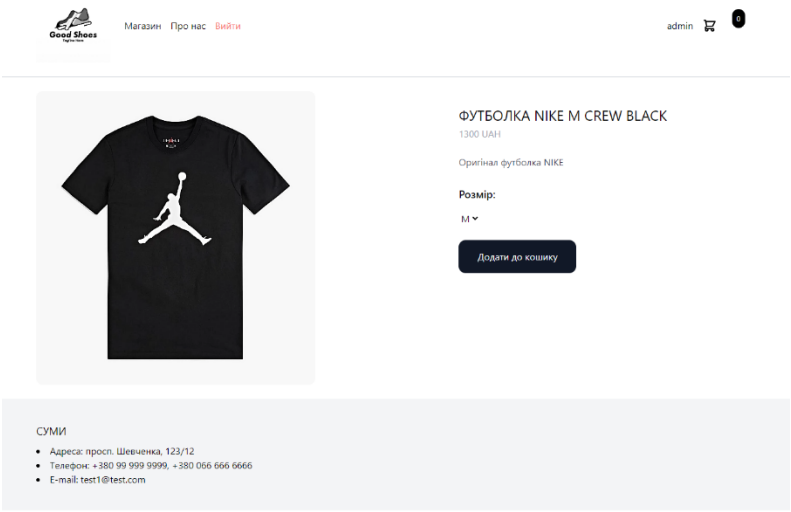


Рисунок 3.17 – Сторінка товару

Сторінка кошик (рис. 3.18) містить інформацію про товар, його ціну, фото, можливість видаляти товар з кошика, змінювати його кількість, а також загальна вартість та кнопка оформлення.

Повний код додатку cart/views.py знаходиться в "Додаток А".

```
def add_to_cart(request, product_id):
    cart = Cart(request)
    cart.add(product_id)
    return render(request, 'cart/cart_menu.html')
def cart(request):
    return render(request, 'cart/cart.html')
def success(request):
    return render(request, 'cart/success.html')
def update_cart(request, product_id, action):
    cart = Cart(request)
    if action == 'increment':
        cart.add(product_id, 1, True)
    else:
        cart.add(product_id, -1, True)
```

```
product = Product.objects.get(pk=product_id)
quantity = cart.get_item(product_id)
if quantity:
    quantity = quantity['quantity']
    item = {
        'product': {
            'id': product.id,
            'name': product.name,
            'image': product.image,
            'get_thumbnail': product.get_thumbnail(),
            'price': product.price,
        },
        'total_price': (quantity * product.price),
        'quantity': quantity,
    }
    response = render(request, 'cart/partials/cart_item.html', {'item': item})
    response['HX-Trigger'] = 'update-menu-cart'
elif action == 'remove':
    cart.remove(product)
    response = render(request, 'cart/partials/cart_item.html')
    response['HX-Trigger'] = 'update-menu-cart'
else:
    item = None
return response
```

add_to_cart(request, product_id) – ця функція додає продукт до кошика користувача;

cart(request) – функція просто відрендерює сторінку cart/cart_menu.html(Додаток Б), яка показує деталі кошика користувача;

success(request) – функція відрендерює сторінку успішного оформлення замовлення (cart/success.html(Додаток Б));

update_cart(request, product_id, action) – ця функція оновлює кількість продукту в кошику користувача;

@login_required(checkout(request)) – функція обробляє оформлення замовлення;

@hx_menu_cart(request) – функція використовується для оновлення меню кошика;

@hx_cart_total(request) – функція використовується для оновлення загальної суми кошика.

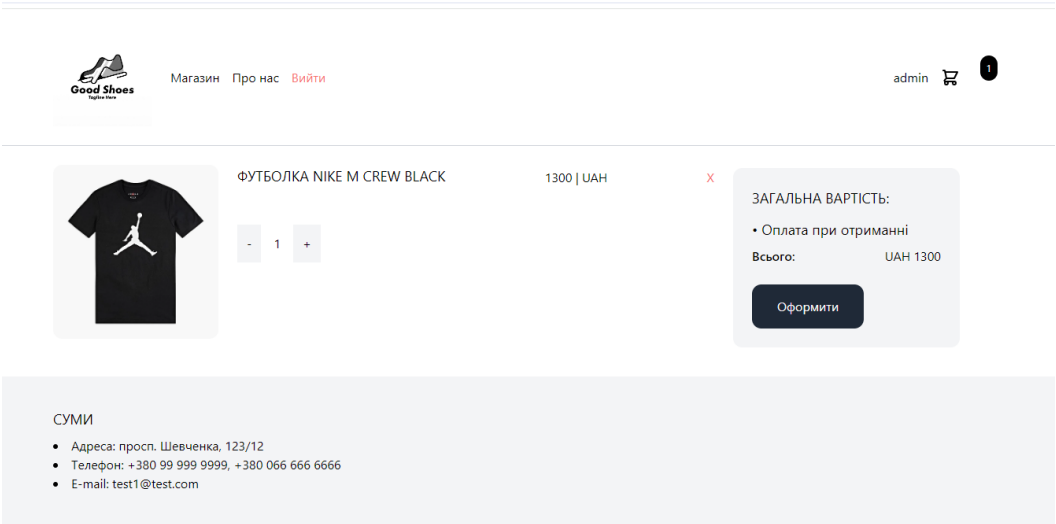


Рисунок 3.18 – Кошик

Сторінка оформлення замовлення (рис. 3.19) містить два блоки. У лівому знаходяться поля які необхідно заповнити для відправки, а справа загальна вартість товару та підтвердження оплати.

Повний код додатку order/views.py знаходиться в "Додаток А".

```
def start_order(request):
    cart = Cart(request)
    data = json.loads(request.body)
```

```
items = []
total_cost = 0
for item in cart:
    product = item['product']
    total_cost += product.price * int(item['quantity'])
    items.append({
        'price_data': {
            'currency': 'usd',
            'product_data': {
                'name': product.name,
            },
            'unit_amount': product.price * 100,
        },
        'quantity': item['quantity']
    })
stripe.api_key = settings.STRIPE_API_KEY_SEC
session = stripe.checkout.Session.create(
    payment_method_types = ['card'],
    line_items=items,
    mode='payment',
    success_url='http://127.0.0.1:8000/cart/success/',
    cancel_url='http://127.0.0.1:8000/cart/'
)
payment_intent = session.payment_intent
order = Order.objects.create(
    first_name = data['first_name'],
    last_name = data['last_name'],
    email = data['email'],
    phone = data['phone'],
```

```
    address = data['address'],
    zipcode = data['zipcode'],
    city = data['city'],
    paid = True,
    paid_amount = total_cost,
)
order.payment_intent = payment_intent
for item in cart:
    product = item['product']
    quantity = int(item['quantity'])
    price = product.price * quantity
    item = OrderItem.objects.create(order=order, product=product, quantity=quantity, price=price)
cart.clear()
return JsonResponse({'session': session, 'order': payment_intent})
```

start_otder(request) – функція обробляє створення замовлення та оплату, використовуючи бібліотеку Stripe для обробки платежів. Шаблон сторінки оформлення замовлення знаходиться в cart/checkout.html(Додаток Б).

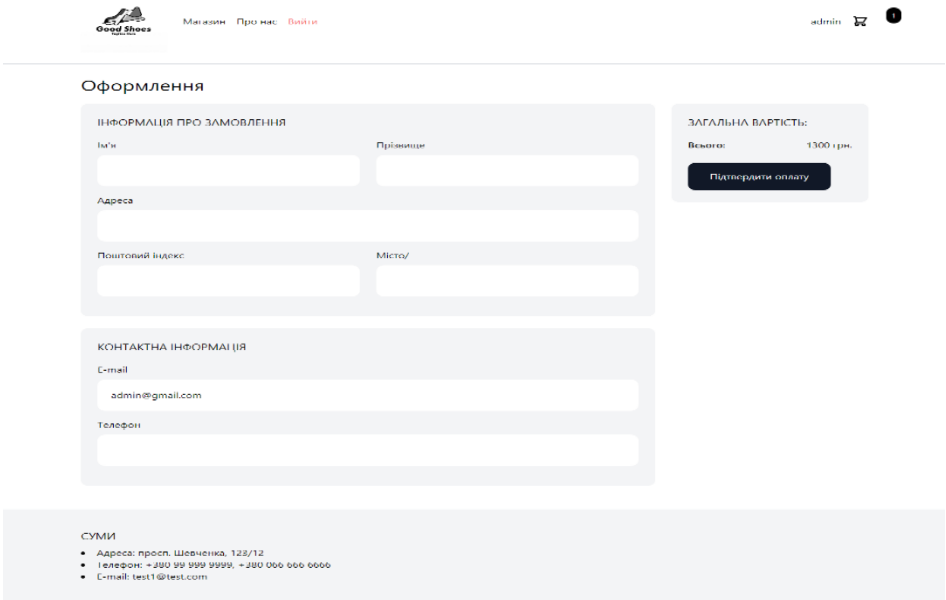


Рисунок 3.19 – Оформлення замовлення

Також розроблено сторінки реєстрації (рис. 3.20) та авторизації (рис. 3.21).

Повний код додатку core/views.py знаходиться в "Додаток А".

```
def signup(request):
    if request.method == 'POST':
        form = SignUpForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('/')
    else:
        form = SignUpForm()
    return render(request, 'core/signup.html', {'form': form})
```

signup(request) – ця функція обробляє реєстрацію користувача. Перевіряє метод HTTP-запиту (request.method). Відрендерює шаблон core/signup.html(Додаток Б).

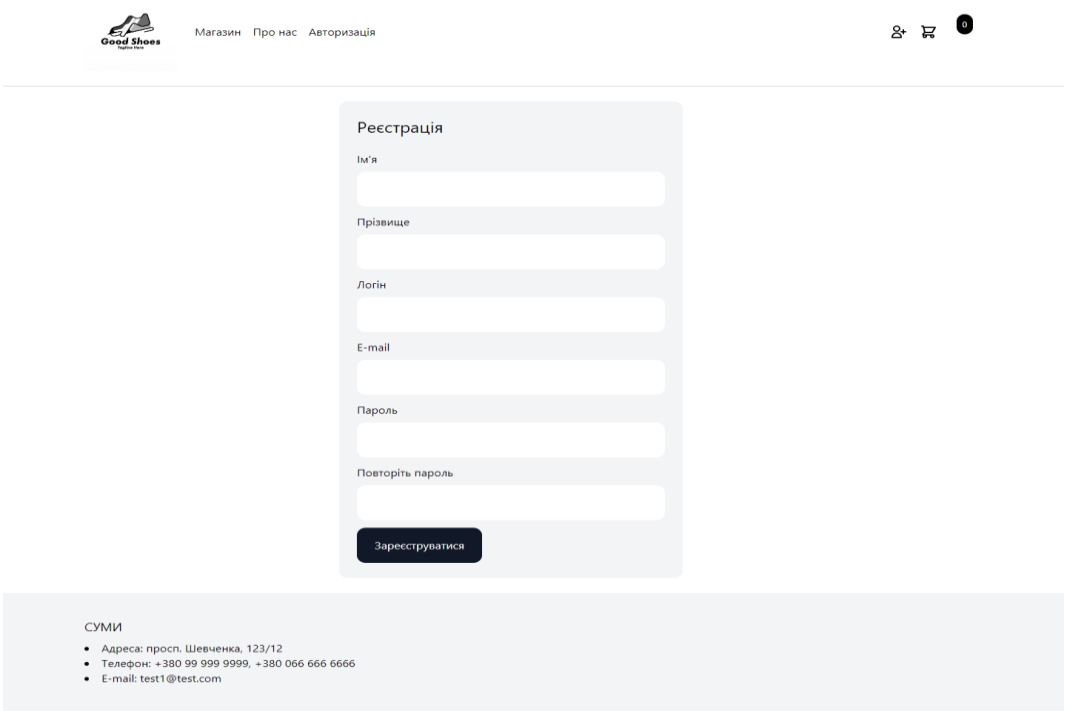


Рисунок 3.20 – Сторінка реєстрації

Шаблон сторінки авторизації core/login.html(Додаток Б).

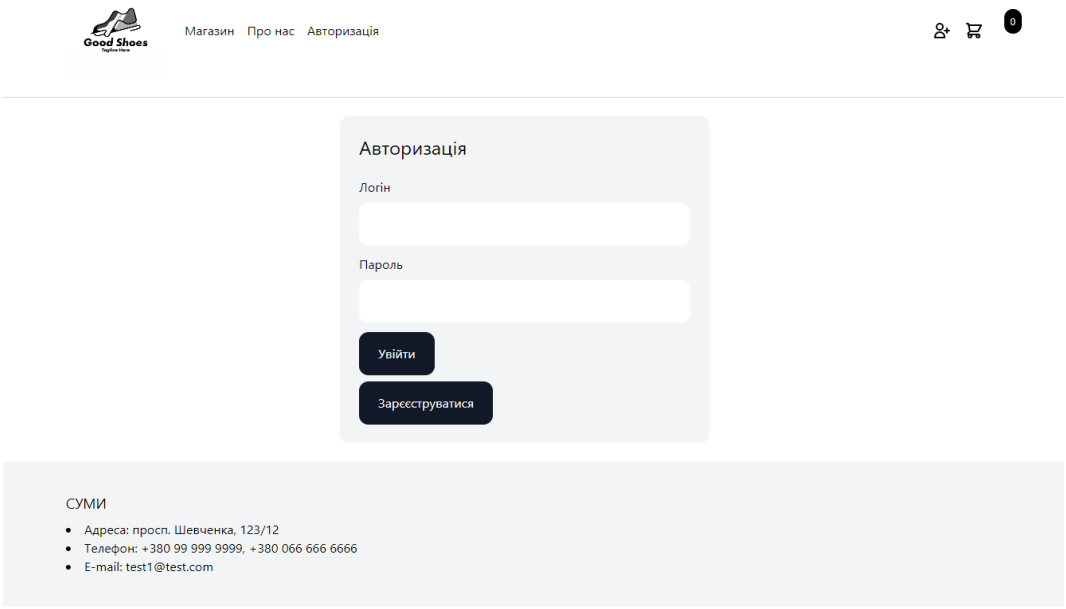


Рисунок 3.21 – Сторінка авторизації

Перейдемо до адміністративної частини сайту:

Адміністративна панель Django (рис. 3.20). Щоб до неї потрапити потрібно авторизуватися на сайті. Доступ до неї мають лише адміністратори.

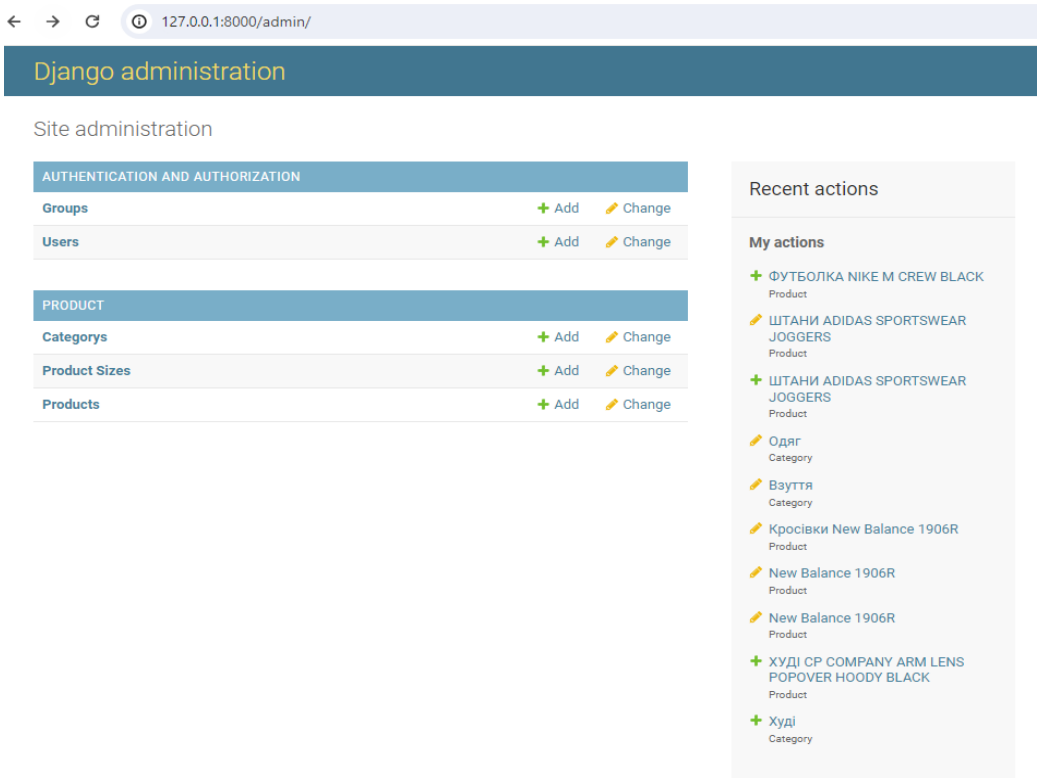


Рисунок 3.20 – Адмін-панель Django

Адміністратор може редагувати користувачів, додавати та видаляти їх, дивитись про них певну інформацію та їх статус.

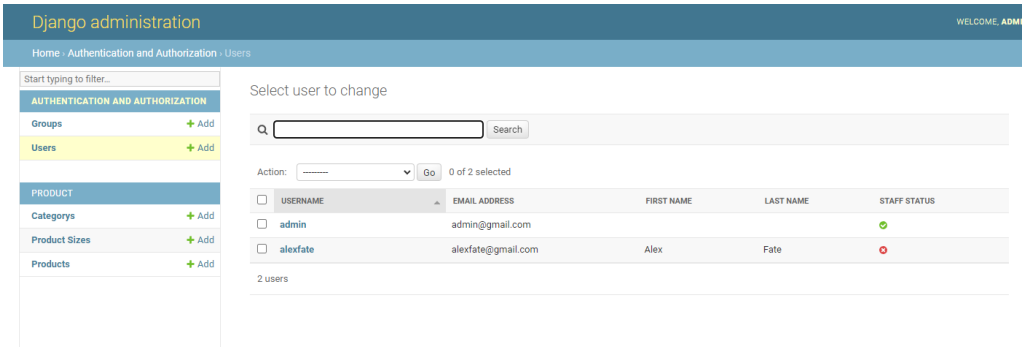


Рисунок 3.21 – Адмін-панель Django

Можливість додавати та видаляти категорії товарів (рис. 3.22), а також переглядати їх (рис. 3.23).

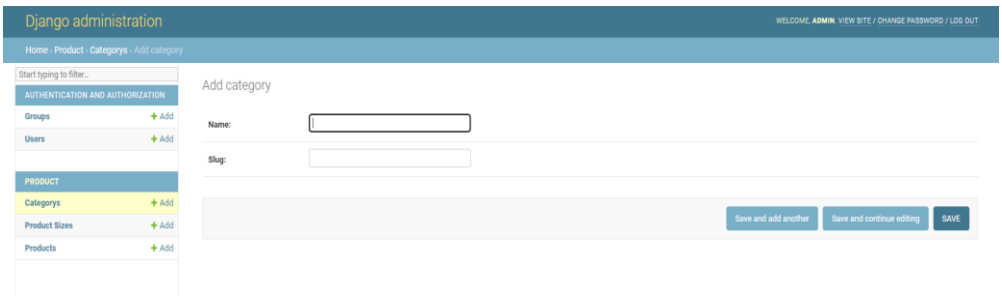


Рисунок 3.22 – Додавання категорій товару

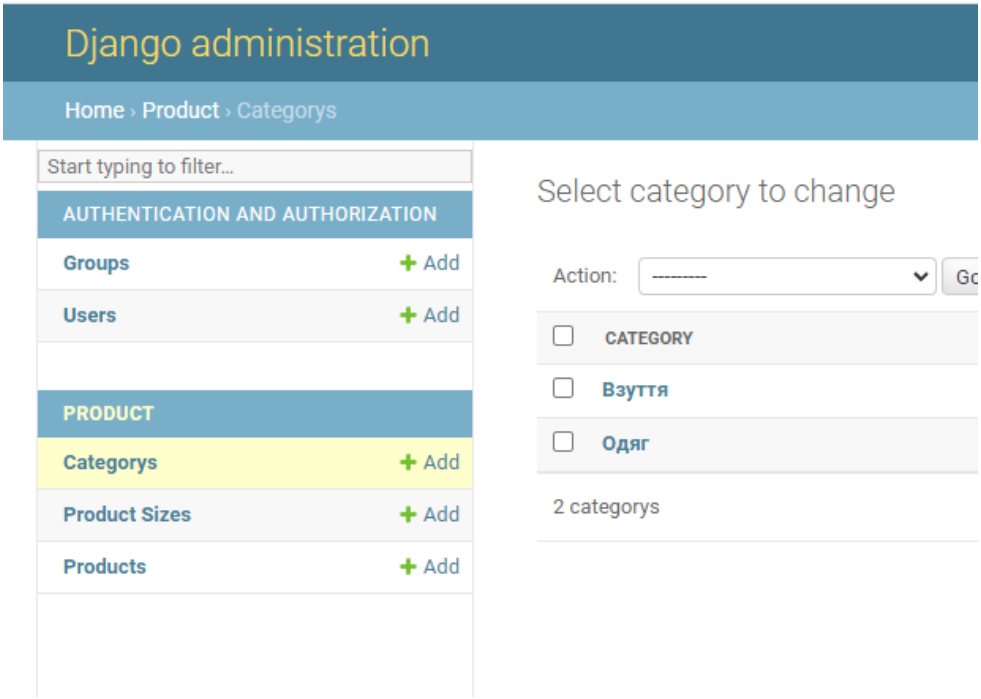


Рисунок 3.23 – Інформація про категорії

Редагувати розміри до товарів (рис. 3.24), переглядати наявність розмірів (рис. 3.25).

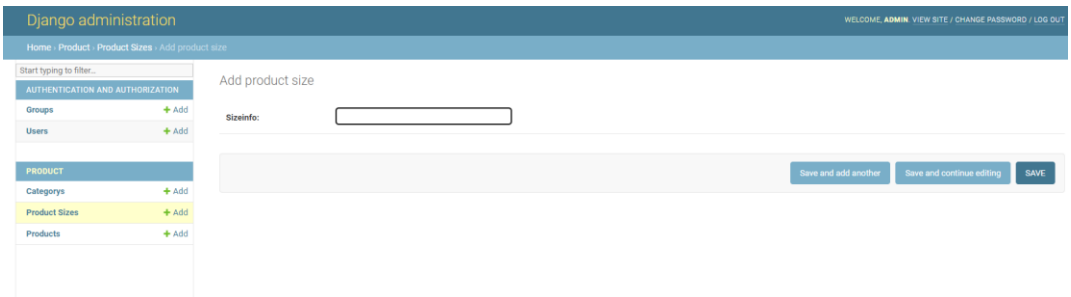


Рисунок 3.24 – Додавання розміру товару

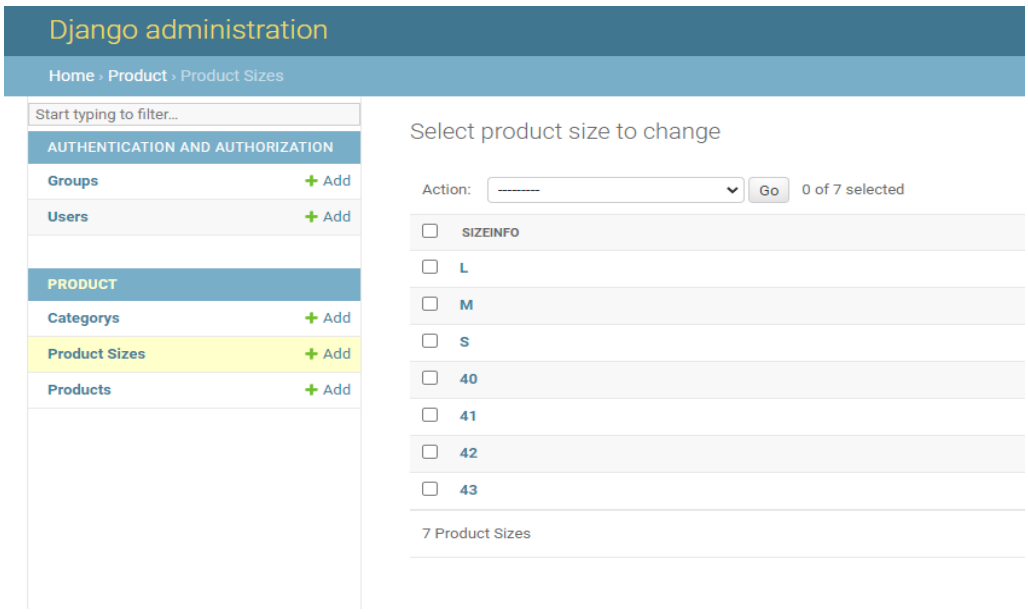


Рисунок 3.25 – Наявні розміри

Додавати/видаляти/редагувати товари: додавати до них опис, ціну, розмір, фото та обирати категорію.

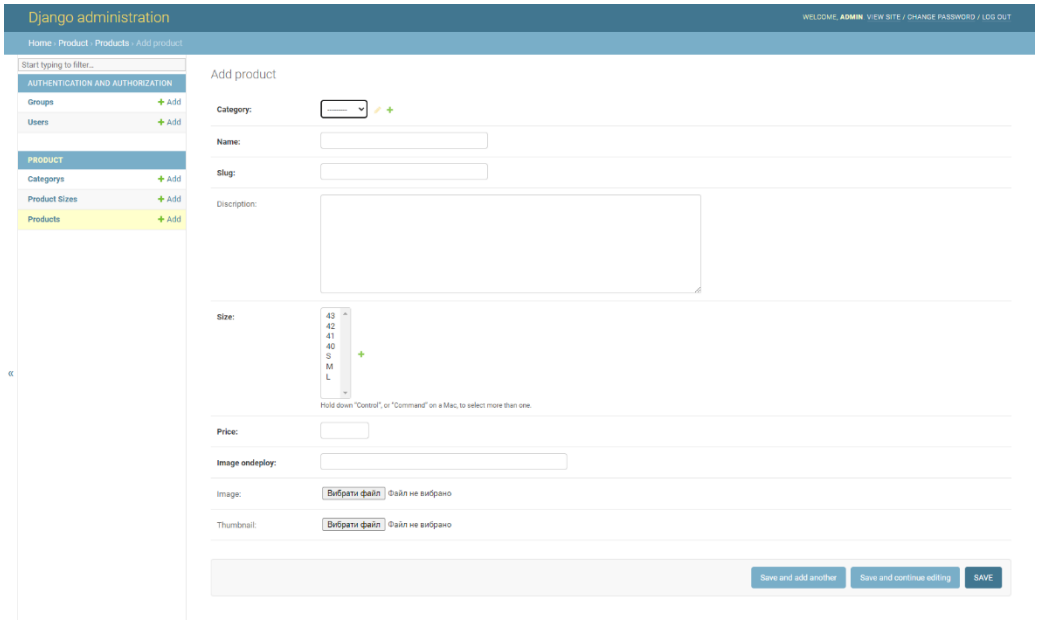


Рисунок 3.26 – Додавання товару

3.4 Тестування

Тестування сайту є важливим процесом для виявлення потенційних помилок на конкретному веб-ресурсі. Перш ніж опублікувати свою веб-систему, ви повинні вжити різні заходи, щоб перевірити вашу роботу на наявність різних помилок і відхилень[16].

Для тестування інтернет-магазину будемо використовувати ручний метод, щоб перевірити чи всі функції виконуються коректно.

Почнемо з головної сторінки, обираємо бажаний товар та додаємо його до кошика (рис. 3.27).

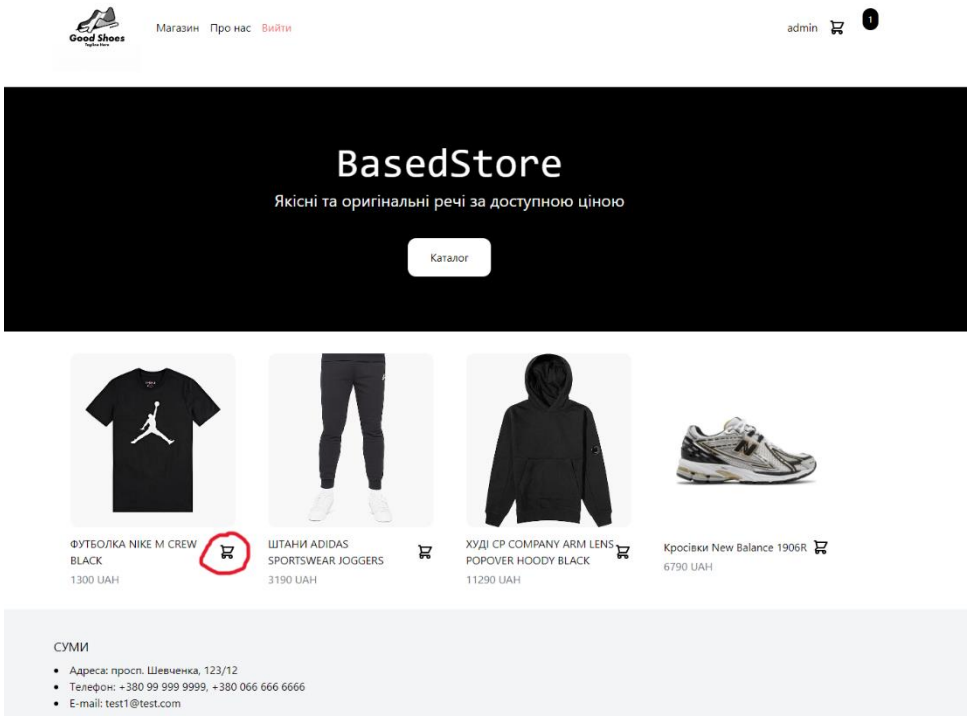


Рисунок 3.27 – Додавання товару до кошика

При натисканні кнопки додати до кошика запит додавання товару в корзину виконаний успішно, про це нас інформує відповідь 200 (рис. 3.28).

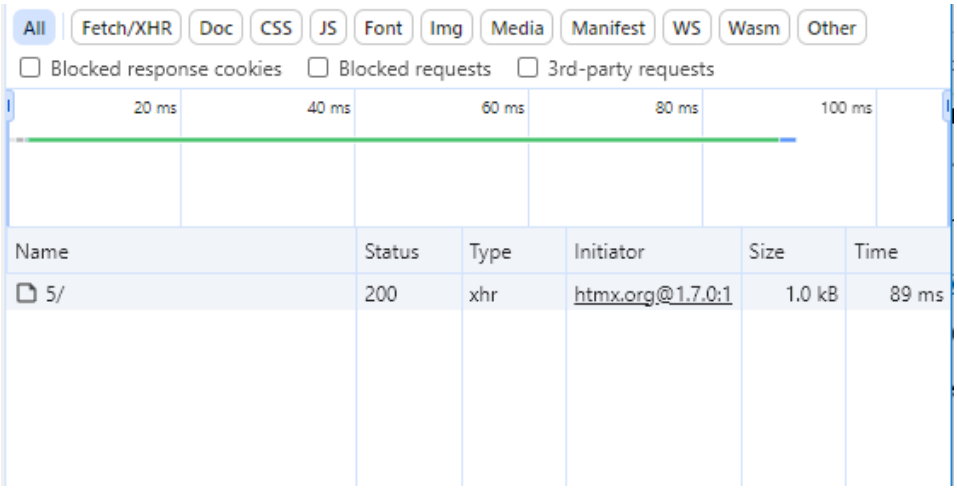


Рисунок 3.28 – Додавання товару до кошика

Видаляємо товар з кошика, натискаємо біля обраного товару кнопку видалення (рис. 3.29).

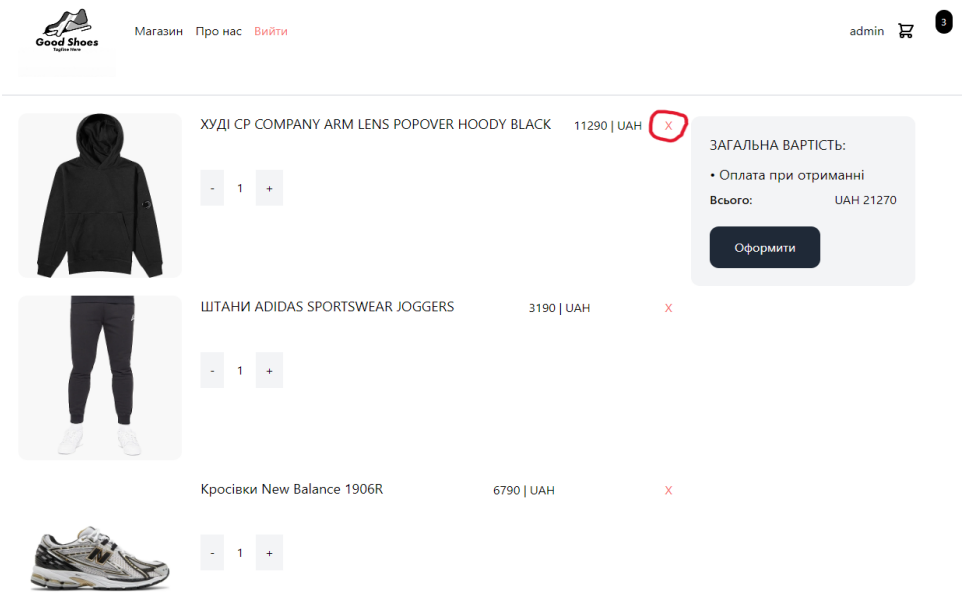
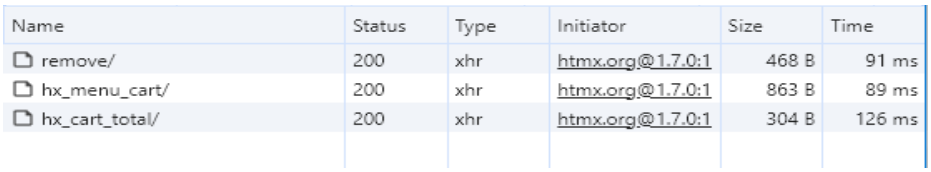


Рисунок 3.29 – Додавання товару до кошика

При натисканні кнопки видалити з кошика запит видалення товару з корзини виконаний успішно, про це нас інформує відповідь 200 (рис. 3.30).



Name	Status	Type	Initiator	Size	Time
remove/	200	xhr	htmx.org@1.7.0:1	468 B	91 ms
hx_menu_cart/	200	xhr	htmx.org@1.7.0:1	863 B	89 ms
hx_cart_total/	200	xhr	htmx.org@1.7.0:1	304 B	126 ms

Рисунок 3.30 – Додавання товару до кошика

Успішне видалення (рис. 3.31).

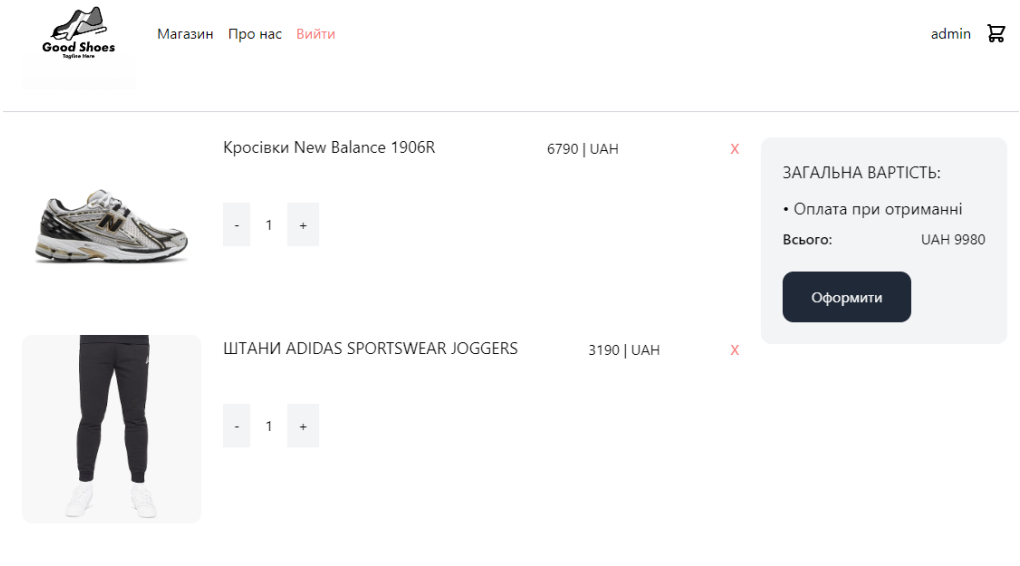


Рисунок 3.31 – Успішний результат видалення

Тепер змінимо кількість товарів у кошику (рис. 3.32). Запит зміни виконано успішно (рис. 3.33).

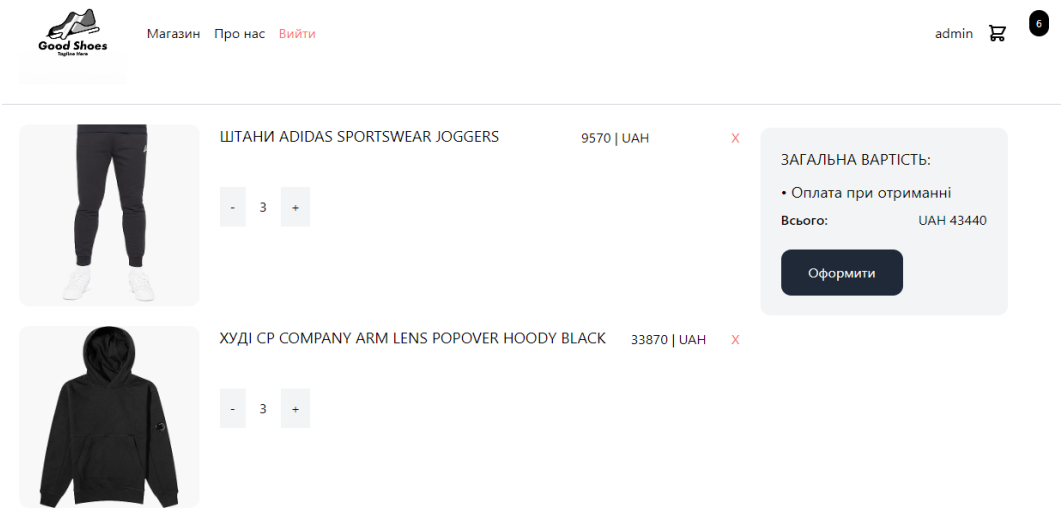


Рисунок 3.32 – Зміна кількості речей у кошику

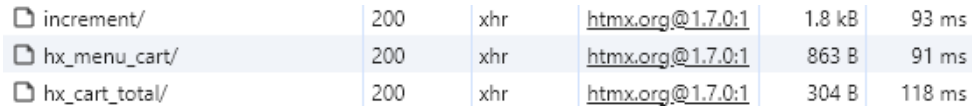


Рисунок 3.33 – Успішна зміна кількості речей

Далі при натисканні кнопки "Оформити" ми маємо отримати повідомлення про успішну покупку. На рисунку 3.34 бачимо підтвердження

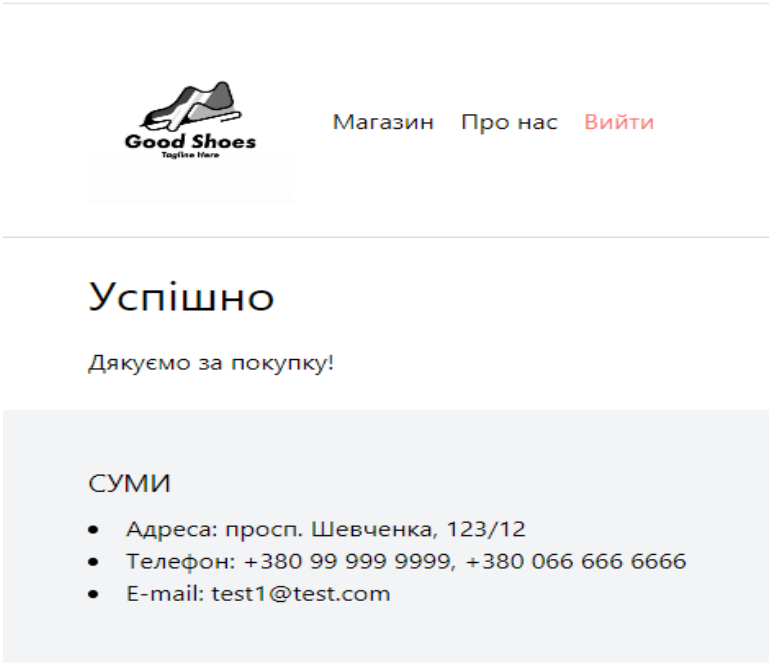


Рисунок 3.34 – Успішне оформлення замовлення

Спробуємо посортувати товар по категорії "Взуття" (рис. 3.35).

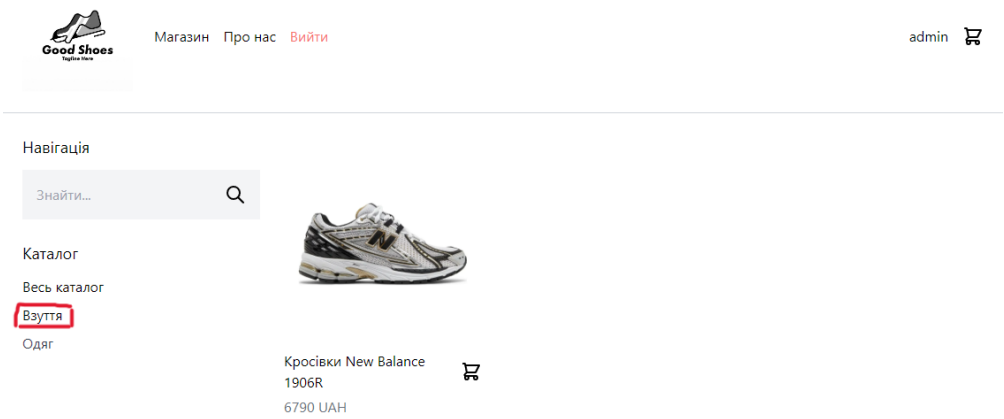


Рисунок 3.35 – Сортування по категорії

При натисканні кнопки запит сортування товару по категорії виконаний успішно, про це нас інформує відповідь 200 (рис. 3.36).

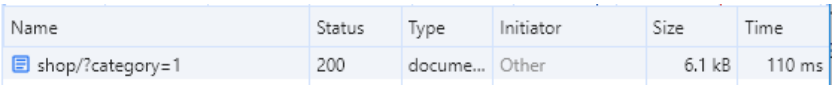


Рисунок 3.36 – Успішне сортування

Протестуємо навігацію, спробуємо знайти товари з назвою "худі" (рис. 3.36). Запит пошуку товару по назві успішно виконано (рис. 3.37).

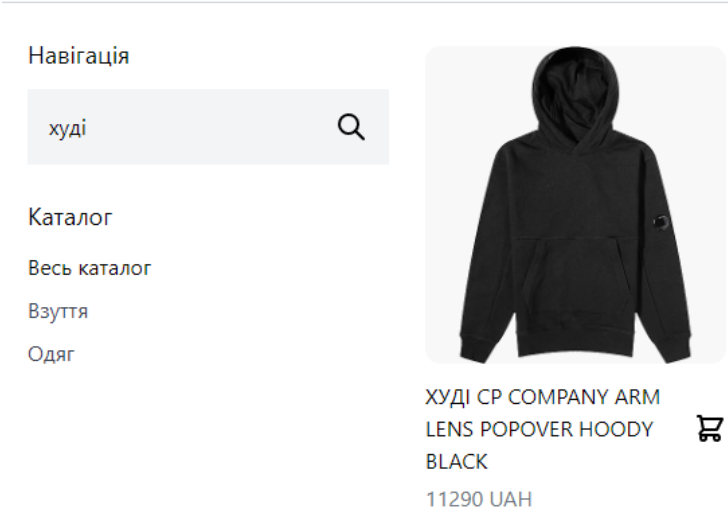


Рисунок 3.36 – Пошук товару по назві

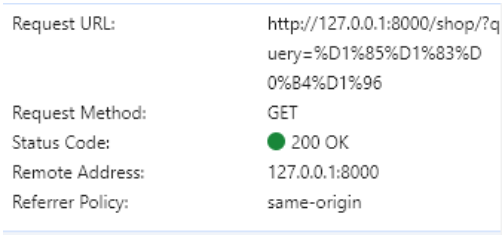


Рисунок 3.37 – Успішний пошук

Було успішно протестовано основний функціонал магазину. Критичних помилок знайдено не було.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи був розроблений інтернет-магазин оригінального одягу та взуття.

- Проведено аналіз аналогічних інтернет-магазинів для виявлення плюсів та недоліків, які враховувалися при створенні сайту.
- Після перегляду усіх популярних фреймворків на мові Python було вирішено використати саме Django, а у якості СУБД було обрано PostgreSQL для зберігання відомостей про товари та замовлення.
- Був розроблений прототип сайту, спроектовано UML, Use Case та ER діаграми.
- Розроблений дизайн є зручним для користувача.
- Реалізовані всі основні функції, такі як: перегляд товарів, навігація, додавання/видалення товарів до кошика, перегляд раніше замовлених товарів, оформлення замовлення тощо.
- Наприкінці проведено тестування веб-сайту, значних помилок не виявлено.

Результатом стала готова платформа для продажу одягу та взуття, яку у майбутньому є можливість удосконалювати та розширювати. Всі поставлені цілі та завдання було виконано.

СПИСОК ЛІТЕРАТУРИ

1. Чим відрізняється маркетплейс від інтернет-магазину? URL: https://allo.ua/ua/chym-vidrizniaietsia-marketpleis-vid-internet-mahazynu_am0-6 (дата звернення: 06.07.2023).
2. Що таке інтернет-магазин? Статті компанії «TDSport - спортивний інтернет-магазин». "TDSport - спортивний інтернет-магазин" - контакти, товари, послуги, ціни. URL: https://tdsport.com.ua/ua/a350577-chto-takoe-internet.html (дата звернення: 06.07.2023).
3. Етапи створення веб сайтів. URL: https://webtune.com.ua/statti/web-rozrobka/etapy-stvorennya-veb-sajtiv (дата звернення: 08.07.2023).
4. Аналіз сайту конкурентів, що варто дізнатися перед створенням власного ресурсу. ™ Глянець – Розробка і підтримка сайтів. URL: https://glyanec.net/ua/blog/analiz-saytu-konkurentiv-scho-varto-diznatisya-pered-stvorennyam (дата звернення: 08.07.2023).
5. Розетка - інтернет-магазин (rozetka.ua) переваги і недоліки. Перший незалежний сайт відгуків України. URL: https://www.otzyvua.net/rozetka-internet-magazin-rozetkaua/plus-i-minus (дата звернення: 08.07.2023).
6. yesoriginal.com.ua інтернет-магазин. Перший незалежний сайт відгуків України | OtzyvUA.net. URL: https://www.otzyvua.net/httpsyesoriginalcomua-httpswwwinstagramcomyes_original (дата звернення: 08.07.2023).
7. Фреймворки Python: Призначення та застосування при розробці ПЗ. FoxmindEd. URL: https://foxminded.ua/ru/frejmvorki-python/ (дата звернення: 11.07.2023).
8. Найкращі фреймворки для Python-розробників та веб-розробки. IT-компанія повного циклу розробки програмних продуктів WEZOM - Київ, Україна. URL: https://wezom.com.ua/blog/python-frameworks (дата звернення: 11.07.2023).

9. Найкращі Бекенд Фреймворки 2024 - Merehead. Merehead. URL: https://merehead.com/ru/blog/best-backend-frameworks-2024/ (дата звернення: 11.07.2023).
10. Що таке PostgreSQL? - Особливості - Переваги і недоліки. Education-WIKI.com. URL: https://uk.education-wiki.com/5154595-what-is-postgresql (дата звернення: 16.04.2024).
11. Все, що потрібно знати про бази даних для початківців: MySQL, PostgreSQL, MongoDB. URL: https://dan-it.com.ua/uk/blog/vse-shho-potribno-znati-pro-bazi-danih-dlja-pochatkivciv-mysql-postgresql-mongodb/#_MySQL_PostgreSQL_MongoDB (дата звернення: 16.04.2024).
12. Прототип сайту, як створення прототипу допомагає оцінити та розробити сайт. Evergreen - web розробка і діджиталізація бізнесу за допомогою AI продуктів. URL: https://evergreens.com.ua/ua/articles/prototype-site.html (дата звернення: 16.05.2024).
13. Моралес Дж. What is UML Diagram: Including UML Diagram Arrow and Symbol. MindOnMap | Free Mind Mapping Tool to Draw Ideas Easily Online. URL: https://www.mindonmap.com/uk/blog/what-is-uml-diagram/ (дата звернення: 18.04.2024).
14. Діаграми UML для моделювання процесів і архітектури проекту. Evergreen - web розробка і діджиталізація бізнесу за допомогою AI продуктів. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 18.04.2024).
15. Реляційні бази даних. Relational databases. URL: https://rdb.dp.ua/uk/chapter_04 (дата звернення: 28.04.2024).
16. Тестування сайту - зайві витрати чи необхідність? URL: https://creator-systems.com/ua/news/testuvannya-saytu-zayvi-vytraty-chy-neobkhidnist (дата звернення: 30.04.2024).

ДОДАТОК А

cart > cart.py

cart > cart.py > Cart > __iter__

```python
from django.conf import settings

from product.models import Product

class Cart(object):

    def __init__(self, request):
        """Ініціалізація сесії та перевірка на існування кошика"""
        self.session = request.session
        cart = self.session.get(settings.CART_SESSION_ID)

        if not cart:
            cart = self.session[settings.CART_SESSION_ID] = {}

        self.cart = cart #Для подальшого використання в інших методах цього класу

    def __iter__(self):
        """Цикл для доступу до Product з бд та збільшення вартості відповідно до кількості продукту"""
        for prod in self.cart.keys():
            self.cart[str(prod)]['product'] = Product.objects.get(pk=prod)

        for item in self.cart.values():
            item['total_price'] = int(item['product'].price * item['quantity'])

            yield item

    def __len__(self):
        """Для вимірювання кількості предметів в кошику"""
        return sum(item['quantity'] for item in self.cart.values())

    def save(self):
        """Оновлення сесії для користувача, при редагуванні кошика"""
        self.session[settings.CART_SESSION_ID] = self.cart
        self.session.modified = True

    def add(self, product_id, quantity=1, update_quantity=False):
        """Додавання в кошик"""

        product_id = str(product_id)

        if product_id not in self.cart:
            self.cart[product_id] = {'quantity': 1, 'id': product_id}

        if update_quantity:
            self.cart[product_id]['quantity'] += int(quantity)

            if self.cart[product_id]['quantity'] == 0:
                self.remove(product_id)

        self.save()

    def remove(self, product_id):
        product_id = str(product_id)
        if product_id in self.cart:
            del self.cart[product_id]

        self.save

    def clear(self):
        del self.session[settings.CART_SESSION_ID]
        self.session.modified = True

    def get_total_cost(self):
        """Отримання всієї вартості кошика"""
        for prod in self.cart.keys():
            self.cart[str(prod)]['product'] = Product.objects.get(pk=prod)
```

```
        return sum(item['product'].price * item['quantity'] for item in self.cart.values())

    def get_item(self, product_id):
        if str(product_id) in self.cart:
            return self.cart[str(product_id)]
        else:
            return None
```

Cart > urls.py

```
cart > urls.py > ...
from django.urls import path

from cart.views import add_to_cart, cart, checkout, hx_menu_cart, hx_cart_total, update_cart, success


urlpatterns = [
    path('', cart, name='cart'),
    path('checkout/', checkout, name='checkout'),
    path('success/', success, name='success'),
    path('hx_menu_cart/', hx_menu_cart, name='hx_menu_cart'),
    path('hx_cart_total/', hx_cart_total, name='hx_cart_total'),
    path('add_to_cart/<int:product_id>/', add_to_cart, name='add_to_cart'),
    path('update_cart/<int:product_id>/<str:action>/', update_cart, name='update_cart'),
]
```

Cart > views.py

cart > views.py > ...

```python
from django.contrib.auth.decorators import login_required
from django.shortcuts import render
from django.conf import settings

from .cart import Cart

from product.models import Product

def add_to_cart(request, product_id):
    cart = Cart(request)
    cart.add(product_id)

    return render(request, 'cart/cart_menu.html')

def cart(request):
    return render(request, 'cart/cart.html')

def success(request):
    return render(request, 'cart/success.html')

def update_cart(request, product_id, action):
    cart = Cart(request)

    if action == 'increment':
        cart.add(product_id, 1, True)
    else:
        cart.add(product_id, -1, True)

    product = Product.objects.get(pk=product_id)
    quantity = cart.get_item(product_id)

    if quantity:
        quantity = quantity['quantity']

        item = {
            'product': {
                'id': product.id,
                'name': product.name,
                'image': product.image,
                'get_thumbnail': product.get_thumbnail(),
                'price': product.price,
            },
            'total_price': (quantity * product.price),
            'quantity': quantity,
        }

        response = render(request, 'cart/partials/cart_item.html', {'item': item})
        response['HX-Trigger'] = 'update-menu-cart'

    elif action == 'remove':
        cart.remove(product)
        response = render(request, 'cart/partials/cart_item.html')
        response['HX-Trigger'] = 'update-menu-cart'
    else:
        item = None

    return response
```

```python
@login_required
def checkout(request):
    pub_key=settings.STRIPE_API_KEY_PUB
    return render(request, 'cart/checkout.html', {'pub_key': pub_key})

def hx_menu_cart(request):
    return render(request, 'cart/cart_menu.html')

def hx_cart_total(request):
    return render(request, 'cart/partials/cart_total.html')

```

Core > forms.py

core > forms.py > ...

```python
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User

class SignUpForm(UserCreationForm):
    email = forms.EmailField(max_length=255, required=True)
    username = forms.CharField(max_length=50, required=True)
    first_name = forms.CharField(max_length=50, required=True)
    last_name = forms.CharField(max_length=50, required=True)
    class Meta:
        model = User
        fields = ['username','email', 'password1', 'password2', 'first_name', 'last_name']
```

Core > urls.py

core > urls.py > ...

```python
from django.urls import path, include
from django.contrib.auth import views

from core.views import shop, frontpage, signup, myaccount, aboutus, edit_myaccount
from product.views import product

urlpatterns = [
    path('', frontpage, name='frontpage'),
    path('aboutus/', aboutus, name='aboutus'),
    path('myaccount/', myaccount, name='myaccount'),
    path('edit_myaccount/', edit_myaccount, name='edit_myaccount'),
    path('login/', views.LoginView.as_view(template_name='core/login.html'), name='login'),
    path('logout/', views.LogoutView.as_view(), name='logout'),
    path('signup/', signup, name='signup'),
    path('shop/<slug:slug>/', product, name='product'),
    path('shop/', shop, name='shop'),


]

```

Core > views.py

```python
from django.contrib.auth import login
from django.contrib.auth.decorators import login_required
from django.db.models import Q
from django.shortcuts import render, redirect

from product.models import Product, Category

from .forms import SignUpForm

def frontpage(request):
    products = Product.objects.all()[0:8]

    return render(request, 'core/frontpage.html', {'products': products})

def aboutus(request):
    return render(request, 'core/aboutus.html')

def signup(request):
    if request.method == 'POST':
        form = SignUpForm(request.POST)

        if form.is_valid():
            user = form.save()

            login(request, user)

            return redirect('/')
    else:
        form = SignUpForm()

    return render(request, 'core/signup.html', {'form': form})

@login_required
def myaccount(request):
    return render(request, 'core/myaccount.html')

@login_required
def edit_myaccount(request):
    """Редагування профіля"""
    if request.method == 'POST':

        user = request.user
        user.first_name = request.POST.get('first_name')
        user.last_name = request.POST.get('last_name')
        user.username = request.POST.get('username')
        user.email = request.POST.get('email')

        user.save()

        return redirect('myaccount')
    return render(request, 'core/edit_myaccount.html')

def shop(request):
    categories = Category.objects.all()
    products = Product.objects.all()
```

```python

    """Активна категорія (Категорія, на якій знаходиться користувач) """
    active_category = request.GET.get('category', '')

    if active_category:
        products = products.filter(category__slug=active_category)

    query = request.GET.get('query', '')
    """Пошук товарів за назвою (в навигації по сайту)"""
    if query:
        products = products.filter(Q(name__icontains=query) | Q(discription__icontains=query))

    """Відображення елементів на сторінці з бази даних"""
    context = {
        'categories': categories,
        'products': products,
        'active_category': active_category,
    }

    return render(request, 'core/shop.html', context)

```

Order > models.py

order > models.py > ...

```python
from itertools import product
from django.contrib.auth.models import User
from django.db import models

from product.models import Product

class Order(models.Model):
    ORDERED = 'ordered'
    SHIPPED = 'shipped'

    STATUS_CHOICES = (
        (ORDERED, 'Ordered'),
        (SHIPPED, 'Shipped')
    )

    first_name = models.CharField(max_length=255)
    last_name = models.CharField(max_length=255)
    address = models.CharField(max_length=255)
    zipcode = models.CharField(max_length=255)
    city = models.CharField(max_length=255)

    email = models.CharField(max_length=255, blank=True)
    phone = models.CharField(max_length=255)

    created_at = models.DateTimeField(auto_now_add=True)

    paid = models.BooleanField(default=False)
    paid_amount = models.IntegerField(blank=True, null=True)

    status = models.CharField(max_length=20, choices=STATUS_CHOICES, default=ORDERED)

    class Meta:
        ordering = ('-created_at',)

    def get_total_price(self):
        if self.paid_amount:
            return self.paid_amount

class OrderItem(models.Model):
    order = models.ForeignKey(Order, related_name='items', on_delete=models.CASCADE)
    product = models.ForeignKey(Product, related_name='items', on_delete=models.CASCADE)
    price = models.IntegerField()
    quantity = models.IntegerField(default=1)

    def get_total_price(self):
        return self.price
```

Order > urls.py

```python
from django.urls import path

from .views import start_order

urlpatterns = [
    path('start_order/', start_order, name='start_order'),
]
```

Order > views.py

```python
import json
import stripe

from django.conf import settings
from django.http import JsonResponse

from cart.cart import Cart

from .models import Order, OrderItem

def start_order(request):
    cart = Cart(request)
    data = json.loads(request.body)

    items = []

    total_cost = 0

    for item in cart:
        product = item['product']
        total_cost += product.price * int(item['quantity'])

        items.append({
            'price_data': {
                'currency': 'usd',
                'product_data': {
                    'name': product.name,
                },
                'unit_amount': product.price * 100,
            },
            'quantity': item['quantity']
        })

    stripe.api_key = settings.STRIPE_API_KEY_SEC
    session = stripe.checkout.Session.create(
        payment_method_types = ['card'],
        line_items=items,
        mode='payment',
        success_url='http://127.0.0.1:8000/cart/success/',
        cancel_url='http://127.0.0.1:8000/cart/'
    )

    payment_intent = session.payment_intent

    order = Order.objects.create(

        first_name = data['first_name'],
        last_name = data['last_name'],
        email = data['email'],
        phone = data['phone'],
        address = data['address'],
        zipcode = data['zipcode'],
        city = data['city'],
        paid = True,
        paid_amount = total_cost,
    )

    order.payment_intent = payment_intent
```

```
59
60
61      for item in cart:
62          product = item['product']
63          quantity = int(item['quantity'])
64          price = product.price * quantity
65
66          item = OrderItem.objects.create(order=order, product=product, quantity=quantity, price=price)
67
68      cart.clear()
69
70      return JsonResponse({'session': session, 'order': payment_intent})
```

Product > models.py

```python
from django import forms
from django.db import models
from django.core.files import File
from django import forms

from PIL import Image
from io import BytesIO

class Category(models.Model):
    """Категорія товарів"""
    name = models.CharField(max_length=255)
    slug = models.SlugField()

    class Meta:
        ordering = ('name', )

    def __str__(self):
        return self.name

class ProductSize(models.Model):
    sizeinfo = models.CharField(max_length=20)

    class Meta:
        verbose_name_plural = 'Product Sizes'

    def __str__(self):
        return self.sizeinfo

class Product(models.Model):

    category = models.ForeignKey(Category, related_name='products', on_delete=models.CASCADE)
    name = models.CharField(max_length=255)
    slug = models.SlugField(unique=True)
    discription = models.TextField(blank=True, null=True)
    size = models.ManyToManyField(ProductSize)
    price = models.IntegerField()
    created_at = models.DateTimeField(auto_now_add=True)
    image_ondeploy = models.URLField(max_length=200, null=True)
    image = models.ImageField(upload_to='uploads/', blank=True, null=True)
    thumbnail = models.ImageField(upload_to='uploads/', blank=True, null=True)

    def get_display_price(self):
        return self.price

    def get_thumbnail(self):
        if self.thumbnail:
            return self.thumbnail.url
        else:
            if self.image:
                self.thumbnail = self.make_thumbnail(self.image)
                self.save()

                return self.thumbnail.url
            else:
                return 'https://via.placeholder.com/240x240x.jpg'

    def make_thumbnail(self, image, size=(300, 300)):
        """Автоматичне створення маленького зображення після завантаження звичайного"""
        img = Image.open(image)
        img.convert('RGB')
        img.thumbnail(size)

        thumb_io = BytesIO()
        img.save(thumb_io, 'JPEG', quality=85)

        thumbnail = File(thumb_io, name=image.name)
```

```
        return thumbnail


    class Meta:
        ordering = ('-created_at', )

    def __str__(self):
        return self.name

```

Product > urls.py

```
product > urls.py > ...
from django.urls import path

from .views import get_size
urlpatterns = [
    path('get_size/<str:product_size>/', get_size, name='get_size'),
]
```

Product > views.py

```
product > views.py > ...
from django.shortcuts import render, get_object_or_404
from .models import Product

def product(request, slug):
    product = get_object_or_404(Product, slug=slug)
    return render(request, 'product/product.html', {'product': product})

def get_size(request, product_size):
    size = Product.objects.get(pk=product_size)

    if request.POST:

        pass

    return render(request, 'product/product.html')
```

Web-app > urls.py

```
web-app > urls.py > ...
from re import template
from django.conf import settings
from django.conf.urls.static import static
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('', include('core.urls')),
    path('cart/', include('cart.urls')),
    path('admin/', admin.site.urls),
    path('order/', include('order.urls')),
    path('product/', include('product.urls')),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

ДОДАТОК Б

frontpage.html

```
{% extends 'core/main.html' %}
{% block content %}
<header class="px-6 py-10 lg:py-20 bg-black">
    <div class="max-w-3xl mx-auto text-center ">
        <p class="font-mono mb-2 text-3xl lg:text-6xl text-white">BasedStore</p>

        <p class="mb-10 text-white lg:text-2xl">Якісні та оригінальні речі за доступною ціною</p>

        <a href="{% url 'shop' %}" class="inline-block px-8 py-4 rounded-xl bg-white text-black">Каталог</a>
    </div>
</header>

<div class="max-w-6xl mx-auto py-2 px-6 xl:px-0">
    <div class="products flex items-center flex-wrap">
        {% for product in products %}
            {% include 'product/partials/item_list_product.html' %}
        {% endfor %}
    </div>
</div>
{% endblock %}
```

shop.html

```
{% extends 'core/main.html' %}
{% block content %}

<div class="max-w-6xl mx-auto flex flex-wrap items-start py-6 px-6 xl:px-0">
    <div class="filters w-full lg:w-1/4">
        <h3 class="mb-3 text-lg">Навігація</h3>

        <form method ="get" action=".">
            <div class="flex">
                <input type="text" name="query" class="p-4 bg-gray-100 border-0" placeholder="Знайти...">

                <button class="p-4 bg-gray-100 border-0">
                    <svg xmlns="http://www.w3.org/2000/svg" class="h-6 w-6" fill="none" viewBox="0 0 24 24" stroke="currentColor" stroke-width="2">
                        <path stroke-linecap="round" stroke-linejoin="round" d="M21 21l-6-6m2-5a7 7 0 11-14 0 7 7 0 0114 0z" />
                    </svg>
                </button>
            </div>
        </form>

        <h3 class="mt-6 mb-3 text-lg">Каталог</h3>

        <ul class="space-y-2">
            <li><a href="{% url 'shop' %}" class="{% if not category.slug == active_category %}text-black{% else %}text-gray-700{% endif %}">Весь каталог</a></li>
            <!-- Категорія товарів з бази даних-->
            {% for category in categories %}
            <li><a href="{% url 'shop' %}?category={{ category.slug }}" class="{% if category.slug == active_category %}text-black{% else %}text-gray-700{% endif %}">{{ category.name }}</a></li>
            {% endfor %}


    </div>
    <div class="products w-full lg:w-3/4 -mt-4 flex items-center flex-wrap">
        {% for product in products %}
            {% include 'product/partials/item_list_product.html' %}
        {% endfor %}
    </div>
</div>
{% endblock %}
```

aboutus.html

```
{% extends 'core/main.html' %}
{% block content %}

<div class="max-w-6xl mx-auto flex flex-wrap items-start py-6 px-6 xl:px-0">
    <h1 class="py-4 text-lg text-black-800 font-semibold">BasedStore</h1>

    <div class="mx-auto">
        <h1 class="py-12 mt-6 mb-6 text-black-800 font-semibold">Шановні клієнти, колеги та партнери!</h1>
        <div class="max-w-6xl mx-auto py-2 px-6 xl:px-0 flex items-center justify-between">
            <h2>Ми - команда ентузіастів моди, яка віддана створенню місця, де кожен може знайти щось особливе для свого гардеробу. Наша
            місія - зробити модний шопінг доступним та захопливим для кожного.

            У нас ви знайдете найактуальніші тенденції моди, які поєднують у собі стиль, комфорт і якість. Ми пропонуємо широкий
            вибір оригінального одягу та взуття для будь-якого смаку та обставин.

            Наша команда завжди на зв'язку, готова відповісти на ваші запитання та надати професійну допомогу у виборі ідеального
            образу. Ми регулярно оновлюємо асортимент, щоб ви завжди мали можливість вибирати серед новинок та виразити свій
            неповторний стиль.

            Наша мета - зробити ваші покупки приємними та легкими, тому ми пропонуємо зручну доставку та гнучку систему оплати.

            Долучайтеся до нашої модної спільноти, де кожен може знайти щось для себе. Дякуємо, що обрали нас!</h2>
        </div>
    </div>
</div>
{% endblock %}
```

myaccount.html

```
{% extends 'core/main.html' %}

{% block title %} Мій профіль {% endblock %}

{% block content %}

<div class="max-w-6xl mx-auto p-6">
    <h1 class="mb-5 text-3xl">Мій профіль</h1>

    <div class="mb-6">
        <strong> Логін: </strong> {{ request.user.username }} <br>
        <strong> E-mail: </strong> {{ request.user.email }} <br>
        <strong> Ім'я: </strong> {{ request.user.first_name }} <br>
        <strong> Прізвище: </strong> {{ request.user.last_name }} <br>
    </div>

    <a href={% url 'edit_myaccount' %}>
        <button class="py-4 px-6 rounded-xl text-white bg-gray-900 hover:bg-black ">Редагувати</button>
    </a>
    <hr>

    <h2 class="my-6 text-xl">Мої замовлення</h2>
    {% comment %}

    <div class="w-full mb-6 p-5 flex flex-wrap bg-gray-100 rounded-xl">
        <div class="mb-5 flex justify-between">
            <a href="#">Номер замовлення: 1 </a>
        </div>
        <!--Замовлення-->
        {% comment %}
        <div class="mb-6 w-full">
            <div class="product mb-6 flex pr-6">
                <a href="#" class="w-1/4">
                    <img class="hover:shadow-lg rounded-xl" src="test.jpg">
                </a>

                <div class="w-3/4 pl-6">
                    <div class="flex justify-between">
                        <a href="#" class="text-lg">Назва продукту</a>

                        <p class="mb-6 pt-1 text-black">3700 грн</p>
                    </div>

                    <hr>

                    <div class="mt-6">
                        Кількість: 1
                    </div>
                </div>

                <div>
                    <p class="text-lg">Дата: 27.04.2222</p>
                </div>
            </div>
        </div>
    </div>
    {% endcomment %}
</div>


{% endblock %}
```

edit_myaccount.html

```
{% extends 'core/main.html' %}

{% block title %}Редагувати профіль{% endblock %}

{% block content %}
<div class="max-w-6xl mx-auto p-6">
    <h1 class="mb-5 text-3xl">Редагувати профіль</h1>

    <form method="post" action="." class="space-y-3">
        {% csrf_token %}

        <div>
            <label>Им'я</label>
            <input name="first_name" value="{{ request.user.first_name }}" type="text" class="w-full mt-2 py-4 px-6 bg-gray-100 rounded-xl">
        </div>

        <div>
            <label>Прізвище</label>
            <input name="last_name" value="{{ request.user.last_name }}" type="text" class="w-full mt-2 py-4 px-6 bg-gray-100 rounded-xl">
        </div>

        <div>
            <label>Логін</label>
            <input name="username" value="{{ request.user.username }}" type="text" class="w-full mt-2 py-4 px-6 bg-gray-100 rounded-xl">
        </div>

        <div>
            <label>E-mail</label>
            <input name="email" value="{{ request.user.email }}" type="email" class="w-full mt-2 py-4 px-6 bg-gray-100 rounded-xl">
        </div>

        <div>
            <button class="py-4 px-6 rounded-xl text-white bg-black">Відправити</button>
        </div>
    </form>
</div>
{% endblock %}
```

cart_menu.html

```
<a href="{% url 'cart' %}" id="menu-cart-button" class="relative text-black hover:text-gray-600">
    <svg xmlns="http://www.w3.org/2000/svg" class="h-6 w-6" fill="none" viewBox="0 0 24 24" stroke="currentColor" stroke-width="2">
        <path stroke-linecap="round" stroke-linejoin="round" d="M3 3h2l.4 2M7 13h10l4-8H5.4M7 13L5.4 5M7 13l-2.293 2.293c-.63.63-.184 1.707.707 1.707H17m0 0a2 2 0 100 4 2 2 0 000-4zm-8 2a2 2 0 11-4 0 2 2 0 014 0z" />
    </svg>

    <span class ="p-2 absolute -top-4 -right-12 bg-black text-white text-xs text-rounded rounded-full">{{cart|length}} </span>
</a>
```

success.html

```
{% extends 'core/main.html' %}

{% block title %}Успішно{% endblock %}

{% block content %}
<div class="max-w-6xl mx-auto flex flex-wrap items-start py-6 px-6 xl:px-0">
    <div class="products w-full lg:w-3/4">
        <h1 class="mb-5 text-3xl">Успішно</h1>

        <p>Дякуємо за покупку!</p>
    </div>
</div>
{% endblock %}
```

checkout.html

```
{% extends 'core/main.html' %}
{% block content %}

<div class="max-w-6xl mx-auto flex flex-wrap items-start py-6 px-6 xl:px-0">
    <div class="products w-full lg:w-3/4">
        <h1 class="mb-5 text-3xl">Оформлення</h1>

        <div class="w-full md:pr-6">
            <div class="mb-6 p-6 bg-gray-100 rounded-xl">
                <h2 class="mb-5 uppercase text-lg">Інформація про замовлення</h2>

                <div class="flex space-x-6">
                    <div class="w-1/2 mb-4">
                        <label class="inline-block mb-2">Ім'я</label>
                        <input name="first_name" type="text" class="w-full p-5 rounded-xl" value="{{ request.user.first_name }}">
                    </div>

                    <div class="w-1/2 mb-4">
                        <label class="inline-block mb-2">Прізвище</label>
                        <input name="last_name" type="text" class="w-full p-5 rounded-xl" value="{{ request.user.last_name }}">
                    </div>
                </div>

                <div class="mb-4">
                    <label class="inline-block mb-2">Адреса</label>
                    <input name="address" type="text" class="w-full p-5 rounded-xl">
                </div>

                <div class="flex space-x-6">
                    <div class="w-1/2 mb-4">
                        <label class="inline-block mb-2">Поштовий індекс</label>
                        <input name="zipcode" type="text" class="w-full p-5 rounded-xl">
                    </div>

                    <div class="w-1/2 mb-4">
                        <label class="inline-block mb-2">Місто/</label>
                        <input name="city" type="text" class="w-full p-5 rounded-xl">
                    </div>
                </div>
            </div>

            <div class="mb-6 p-6 bg-gray-100 rounded-xl">
                <h2 class="mb-5 uppercase text-lg">Контактна інформація</h2>

                <div class="mb-4">
                    <label class="inline-block mb-2">E-mail</label>
                    <input name="email" type="email" class="w-full p-5 rounded-xl" value="{{ request.user.email }}">
                </div>

                <div class="mb-4">
                    <label class="inline-block mb-2">Телефон</label>
                    <input name="phone" type="phone" class="w-full p-5 rounded-xl">
                </div>
            </div>
        </div>
    </div>

    <div class="summary w-full md:w-1/4 p-6 bg-gray-100 rounded-xl mt-14">
        <h2 class="uppercase text-lg mb-5">Загальна вартість:</h2>

        <div class="mb-6 flex justify-between">
            <span class="font-semibold">Всього:</span>
            <span>{{ cart.get_total_cost }} грн.</span>
        </div>
```

```
        <button
            onclick="Ultrabuy(event)"
            class="inline-block px-8 py-4 rounded-xl bg-gray-900 hover:bg-black text-white"
        >
            Підтвердити оплату
        </button>

    </div>
</div>
{% endblock %}
```

signup.html

```
{% extends 'core/main.html' %}
{% block content %}

<div class="max-w-lg mx-auto flex flex-wrap p-6">
    <div class="w-full bg-gray-100 p-6 rounded-xl">
        <h1 class="mb-6 text-2xl">Реєстрація</h1>

        <form method="post" action="." class="space-y-3">
            {% csrf_token %}
            <div>
                <label>Ім'я</label>
                <input type="text" name="first_name" class="w-full mt-2 py-4 px-6 bg-white rounded-xl">
            </div>

            <div>
                <label>Прізвище</label>
                <input type="text" name="last_name" class="w-full mt-2 py-4 px-6 bg-white rounded-xl">
            </div>

            <div>
                <label>Логін</label>
                <input type="text" name="username" class="w-full mt-2 py-4 px-6 bg-white rounded-xl">
            </div>

            <div>
                <label>E-mail</label>
                <input type="email" name="email" class="w-full mt-2 py-4 px-6 bg-white rounded-xl">
            </div>

            <div>
                <label>Пароль</label>
                <input type="password" name="password1" class="w-full mt-2 py-4 px-6 bg-white rounded-xl">
            </div>

            <div>
                <label>Повторіть пароль</label>
                <input type="password" name="password2" class="w-full mt-2 py-4 px-6 bg-white rounded-xl">
            </div>

            {% if form.errors %}
                {% for field in form %}
                    {% for error in field.errors %}
                        <div class="p-6 bg-red-200 text-red-800 rounded-xl">
                            <p>{{ error|escape }}</p>
                        </div>
                    {% endfor %}
                {% endfor %}

                {% for error in form.non_field_errors %}
                    <div class="p-6 bg-red-200 text-red-800 rounded-xl">
                        <p>{{ error|escape }}</p>
                    </div>
                {% endfor %}
            {% endif %}

            <div>
                <button class="py-4 px-6 rounded-xl text-white bg-gray-900 hover:bg-black ">Зареєструватися</button>
            </div>

        </form>
    </div>
</div>
{% endblock %}
```

login.html

```
{% extends 'core/main.html' %}
{% block content %}

<div class="max-w-lg mx-auto flex flex-wrap p-6">
    <div class="w-full bg-gray-100 p-6 rounded-xl">
        <h1 class="mb-6 text-2xl">Авторизація</h1>

        <form method="post" action="." class="space-y-3">
            {% csrf_token %}
            <div>
                <label>Логін</label>
                <input type="text" name="username" class="w-full mt-2 py-4 px-6 bg-white rounded-xl">
            </div>

            <div>
                <label>Пароль</label>
                <input type="password" name="password" class="w-full mt-2 py-4 px-6 bg-white rounded-xl">
            </div>

            {% if form.errors %}
                {% for field in form %}
                    {% for error in field.errors %}
                        <div class="p-6 bg-red-200 text-red-800 rounded-xl">
                            <p>{{ error|escape }}</p>
                        </div>
                    {% endfor %}
                {% endfor %}

                {% for error in form.non_field_errors %}
                    <div class="p-6 bg-red-200 text-red-800 rounded-xl">
                        <p>{{ error|escape }}</p>
                    </div>
                {% endfor %}
            {% endif %}

            <div>
                <button class="py-4 px-6 rounded-xl text-white bg-gray-900 hover:bg-black ">Увійти</button>
            </div>

        </form>
        <a href="{% url 'signup' %}">
            <button class="mt-2 py-4 px-6 rounded-xl text-white bg-gray-900 hover:bg-black ">Зареєструватися</button>
        </a>
    </div>
</div>
{% endblock %}
```