КВАЛІФІКАЦІЙНА РОБОТА

Група ІПЗс-20
Савчук В.Р.

2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

на правах рукопису

Савчук Володимир Романович

УДК 004.4

Розробка веб-платформи для продажу будівельних матеріалів

Спеціальність 121 – «Інженерія програмного забезпечення»

Кваліфікаційна робота на здобуття кваліфікації бакалавр

Нормоконтроль

_____Стисло О.В.

(підпис, дата, розшифрування підпису)

Студент

_____Савчук В.Р.

(підпис, дата, розшифрування підпису)

Допускається до захисту

Завідувач кафедри

_____к.т.н., доц. Ващишак С.П.

(підпис, дата, розшифрування підпису)

Керівник роботи

к.т.н., проф. каф. ІТ

_____ Пашкевич О.П.

(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2024

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «бакалавр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

«____» _____ 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Савчук Володимир Романович

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи:
Розробка веб-платформи для продажу будівельних матеріалів

керівник роботи:
Пашкевич Олег Петрович, к.т.н., проф. каф. ІТ

затверджена наказом вищого навчального закладу від « 12 » березня 2024 року № 19/1

2. Термін подання студентом роботи 05.06.2024
3. Вихідні дані роботи: Python, Django, HTML, CSS, Stripe, TailwindCSS, SQL

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)
   1. Аналіз наявних аналогів
   2. Розробка прототипу сайту
   3. Реалізація функціоналу для покупки товару

5. Дата видачі завдання 14.03.2024

КОНСУЛЬТАНТИ РОЗДІЛІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Розділ	Консультант (прізвище, ініціали та посада)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	Огляд та аналіз існуючих аналогів	24.03.2024	Виконано
2.	Проектування прототипу сайту	27.03.2024	Виконано
3.	Розробка сайту	01.04.2024	Виконано
4.	Оформлення пояснювальної записки	25.05.2024	Виконано
5.	Оформлення графічного матеріалу та підготовка до захисту роботи	30.05.2024	Виконано

Студент _____ Савчук В.Р.

(підпис) (прізвище та ініціали)

Керівник роботи _____ Пашкевич О.П.

(підпис) (прізвище та ініціали)

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

АНОТАЦІЯ

У ході кваліфікаційної роботи був розроблений веб-сайт інтернет-магазину, який обладнаний рядом функцій: можливість здійснення покупок, перегляд каталогу, сортування товарів за різними параметрами, особистий кабінет користувача та кошик.

У першому розділі нашої роботи був проведений аналіз конкурентів і аналогів, під час якого були виявлені переваги та недоліки їх функціональності та дизайну. Це дало змогу зробити відповідні висновки для подальшого проектування сайту.

У другому розділі надано детальну інформацію про структуру сайту, розроблено його вигляд з урахуванням запланованих функцій та результатів аналізу, проведеного у першому розділі.

Третій розділ присвячений реалізації веб-сайту інтернет-магазину з використанням таких технологій, як Python, HTML, CSS, JavaScript, а також фреймворків Django і Tailwind.

КЛЮЧОВІ СЛОВА: PYTHON, HTML, CSS, DJANGO, TAILWINDCSS.

SUMMARY

In the course of the thesis, I developed an online store website that is equipped with a number of functions: the ability to make purchases, browse the catalog, sort goods by various parameters, a user account, and a shopping cart.

In the first chapter of our work, we analyzed competitors and analogs, identifying the advantages and disadvantages of their functionality and design. This allowed us to draw relevant conclusions for further website design.

The second section provides detailed information about the structure of the website, and develops its appearance taking into account the planned functions and the results of the analysis conducted in the first section.

The third section is devoted to the implementation of the online store website using technologies such as Python, HTML, CSS, JavaScript, as well as Django and Tailwind frameworks.

KEYWORDS: PYTHON, HTML, CSS, DJANGO, TAILWINDCS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HTML – Hyper Text Markup Language

CSS – Cascading Style Sheets

JS – Java Script

Django – Фреймворк для розробки веб систем

SVG – Scalable Vector Graphics

CDN – Content Delivery Network

SQL – Structured query language

ВСТУП

Актуальність теми. Головною тенденцією в розвитку інформаційних технологій на сьогодні є розвиток Інтернету, вдосконалення його можливостей задля задоволення потреб користувачів зокрема шляхом створення великої кількості різноманітних користувацьких веб-застосувань з використанням новітніх засобів розробки веб застосунків.

Широкого використання можливостей Інтернету набула торгівельна сфера – продажі онлайн нищівно забирають долю ринку «наземних» продаж. Електронні магазини потребують значно менших витрат на утримання та організацію роботи, оскільки у ньому значно обмеженіша матеріально–технічна база (будівлі, споруди, приміщення) та кількість обслуговуючого персоналу, а також паралельно з тим може працювати простий offline магазин, так розширюватися ринки збуту, та можливості покупця: купувати будь-який товар в будь-який час в будь-якій країні в будь-якому місті. Також перевагою інтернет магазину можна вважати його компактність: необхідна наявність одного лише складу в якому буде розташовуватися продукція. Головною причиною, що спонукає людину займатися комерційною діяльністю в мережі інтернет, це низький стартовий капітал і невисокі ризики. Все це надає веб-магазинам перевагу перед звичайними offline магазинами, це прискорює бізнес–процеси за рахунок їх проведення електронним чином. В даному випадку інформація передається безпосередньо до одержувача, обходячи стадію створення паперової копії на кожному етапі. В такому випадку, електронну комерцію можна описати як ведення бізнесу через інтернет. У сучасному суспільстві все більше і більше компаній переходять у всесвітню мережу.

Мета роботи. Розробка онлайн магазину будівельних матеріалів, будівельних інструментів , побутової хімії та інших товарів , з можливістю дізнатися потрібну інформацію про товар , добавити його в корзину і оформити та відслідковувати замовлення.

Об’єкт роботи. Інтернет магазин для продажу будівельних матеріалів.

Предмет роботи. Створення інтернет магазину з особистим кабінетом користувача , каталогом, можливістю покупки товару.

Завдання роботи. Відповідно до обраної теми в роботі покладені такі задачі як:

– пошук та аналіз уже існуючих застосунків аналогічного призначення;

– вибір мови програмування та технологій розробки;

– розроблення дизайну відповідно до тематики сайту;

– розробка сайту з потрібним функціоналом і дизайном;

– проведення тестування продукту.

Методи роботи. Для вирішення поставленого завдання були використані мова програмування Python, база даних PostgreSQL, фреймворк Django.

Результати роботи. Результатом кваліфікаційної роботи є веб платформи по продажу будівельних матеріалів. Веб платформа реалізує функціонал :

– додавання на сайт товарів з різноманітними характеристиками відповідно до категорії;

– пошук товарів за назвою, описом, ціною, виробником;

– додавання товарів кошик;

– покупка товарів та оплата за допомогою платіжної системи Stripe;

– робота з особистим кабінетом користувача.

Апробація результатів дослідження. Матеріали кваліфікаційної роботи були представлені на XI Міжнародна наукова конференція "Студентські наукові дискусії поза форматом", яка відбулась 11 квітня 2024 року в Університеті Короля Данила.

Структура роботи. Розділи – 3. Загальний обсяг основної частини –77 сторінок. Список використаних джерел – 20.

РОЗДІЛ 1. АНАЛІЗ ІНТЕРНЕТ МАГАЗИНІВ

1.1 Потреби бізнесу в веб сайті

З кожним роком кількість користувачів серед населення України, які активно використовують Інтернет, збільшується, і ця тенденція робить мережу невід'ємною частиною життя. Сучасному користувачеві важливо мати доступ до повної інформації про товар чи послугу онлайн. Це особливо відчутно у сфері торгівлі, де офлайн бізнесам дедалі важче конкурувати з онлайн. Згідно з дослідженнями, близько 82% українців використовують інтернет хоча б раз на тиждень, а приблизно 72% проводять в мережі більше 4 годин на день (рис 1.1).

Рисунок 1.1 – Результат опитування

Чимало магазинів вирішують відкрити свої веб-сайти, щоб не пропустити можливості привернути нових покупців. Інтернет-магазинам потрібно значно менше витрат на відкриття та утримання, оскільки їм не потрібні фізичні

приміщення, касові апарати чи великий штат працівників. Крім того, вони мають доступ до різноманітних інструментів цифрового маркетингу, таких як пошукова оптимізація, контекстна реклама, соціальні медіа тощо, що можуть допомогти привернути більше клієнтів та збільшити продажі. Однією з ключових переваг інтернет-магазинів є можливість розміщувати значно більше товарів та послуг, ніж це можливо на прилавках фізичного магазину. Також вони можуть надати більше інформації про товари та послуги, що дозволяє покупцям зробити більш обдуманий вибір.

Загалом, перехід офлайн магазину в онлайн сегмент може принести ряд переваг, таких як розширення аудиторії, збільшення зручності для клієнтів та можливість збільшити обсяги продажів. Однак, існують певні недоліки, такі як :

- потреба в додаткових ресурсах: Управління як фізичним, так і онлайн-відділеннями може вимагати додаткових ресурсів, таких як персонал, програмне забезпечення, технічна підтримка та інше;
- необхідність інтеграції систем: Для забезпечення ефективної роботи магазину як в онлайн, так і офлайн сегментах, необхідно мати інтегровану систему управління, яка об'єднує облік товарів, замовлень, клієнтських даних тощо;
- конкуренція з онлайн-магазинами: Фізичний магазин, що переходить в онлайн, конкурує з великою кількістю інших інтернет-магазинів, які можуть мати більшу експертизу в онлайн-торгівлі та більший бюджет на маркетинг.

Для того щоб інтернет-магазин був успішним він повинен бути невідомим , сайт має відповідати вимогам користувачів . Основними вимогами до інтернет-магазину називають :

- зручність користування;
- простий та інтуїтивно зрозумілий інтерфейс;
- швидке завантаження сторінок;
- зручна навігація та пошук товарів;
- можливість фільтрувати та сортувати товари;

- детальні описи та якісні фото товарів;
- зручна форма замовлення;
- безпечні способи оплати;
- відгуки інших покупців;
- швидкість та якість роботи служби підтримки;
- зручна мобільна версія сайту або мобільний додаток;
- можливість оформляти замовлення з мобільного телефону.

Важливо зазначити, що вимоги користувачів до інтернет-магазинів постійно зростають. Тому, щоб бути успішним, інтернет-магазин повинен не лише відповідати основним вимогам, але й постійно вдосконалюватися, пропонуючи своїм користувачам нові послуги та можливості.

1.2 Критерії вибору будівельний матеріалів

Зважаючи на велику важливість вибору правильних будівельних матеріалів для будь-якого будівельного проекту, покупці дуже ретельно аналізують кожен аспект і параметр матеріалів. Давайте розглянемо кожен з цих критеріїв більш детально:

Якість і довговічність: покупці шукають матеріали, які забезпечують високу якість і мають довгий термін служби. Вони оцінюють міцність, стійкість до впливу погодних умов, ультрафіолетового випромінювання, корозії, а також відновлюваність матеріалів.

Енергоефективність: у сучасних будівлях важливо використовувати енергоефективні матеріали, які допомагають зменшити витрати на опалення та кондиціонування повітря. Покупці оцінюють теплопровідність, герметичність, а також можливість використання альтернативних джерел енергії, таких як сонячні панелі.

Екологічність: зростає попит на екологічно чисті матеріали, які не містять шкідливих речовин і не завдають шкоди довкіллю. Покупці оцінюють вміст токсичних речовин, емісію CO2 під час виробництва та переробки, а також

можливість вторинного використання та переробки.

Ціна: ціновий фактор грає важливу роль у виборі будівельних матеріалів. Покупці шукають оптимальне співвідношення ціни і якості, а також розглядають загальні витрати на матеріали під час експлуатації будівлі.

Доступність і легкість монтажу: матеріали повинні бути легко доступними та встановлюватися без зайвих труднощів. Покупці оцінюють можливість здійснення самостійного монтажу, наявність необхідних інструментів та матеріалів, а також час, необхідний для монтажу.

Естетичний вигляд: важливо, щоб будівельні матеріали відповідали дизайну будівлі та задовольняли естетичні вимоги покупців. Вони оцінюють текстуру, кольори, форми та інші дизайнерські особливості матеріалів.

Технічні характеристики: Покупці також ретельно вивчають технічні характеристики матеріалів, такі як міцність, водонепроникність, маса, розміри, стійкість до вогню та хімічних речовин.

Гарантія і підтримка від виробника: важливо мати гарантію від виробника та підтримку в разі виникнення проблем або несправностей з матеріалами.

Доставка: покупці оцінюють процес доставки матеріалів, включаючи доступність доставки у їхній регіон, час доставки, вартість та умови доставки, а також можливість відстеження замовлення. Важливо, щоб доставка була швидкою, надійною та ефективною, щоб уникнути затримок у будівельних проектах і забезпечити своєчасне отримання матеріалів.

Загальний вибір будівельних матеріалів зазвичай базується на компромісі між всіма цими критеріями, враховуючи конкретні потреби, вимоги та можливості покупців.

1.3 Аналіз конкурентів

Перед створення будь якого продукту потрібно дослідити конкурентів. Це важливий етап розробки сайту оскільки він дозволяє оцінити що дозволяє сайтам в цій сфері бути успішним , які в них є переваги на які переваги треба

зважати і які недоліки не потрібно повторювати щоб інтернет магазин був успішним. Для того щоб розуміти що потрібно від інтернет магазину я вирішив дослідити таких конкурентів як : Rozetka (рис. 1.2), Епіцентр (рис. 1.3) та Нова Лінія (рис. 1.4).

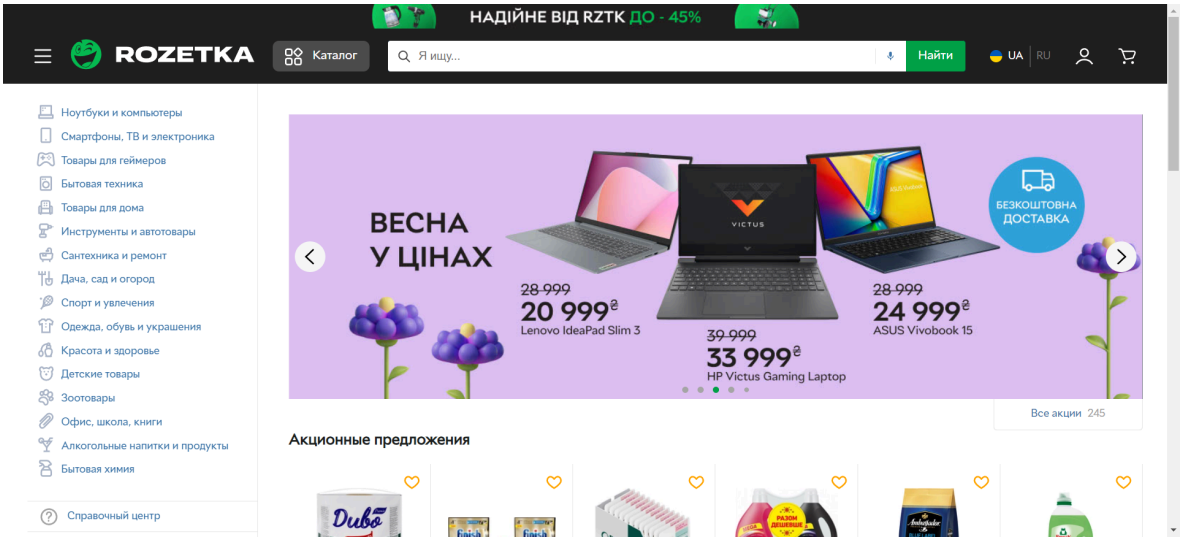


Рисунок 1.2 – Головна сторінка сайту rozetka.com.ua

Переваги:

– широкий асортимент товарів: Rozetka пропонує широкий вибір будівельних матеріалів та інструментів, а також широкий спектр інших товарів, що робить її зручним місцем для комплектування замовлення;

– доступні ціни: Rozetka часто пропонує конкурентні ціни на будівельні матеріали, а також акції та розпродажі, що дозволяє покупцям економити;

– зручний сайт: сайт Rozetka зручний для навігації та пошуку потрібних товарів;

– швидка доставка: Rozetka пропонує швидку доставку по всій Україні;

– можливість повернення товару: Rozetka пропонує можливість повернення товару протягом 14 днів.

Недоліки:

– невеликий асортимент спеціалізованих будівельних матеріалів: Rozetka більше фокусується на товарах для дому та ремонту, а не на спеціалізованих будівельних матеріалах;

– не завжди є професійна консультація щодо будівельних матеріалів: Rozetka не завжди може надати професійну консультацію щодо будівельних матеріалів;

– немає фізичних відділень де можна прийти і отримати консультацію, задати питання і одразу купити;

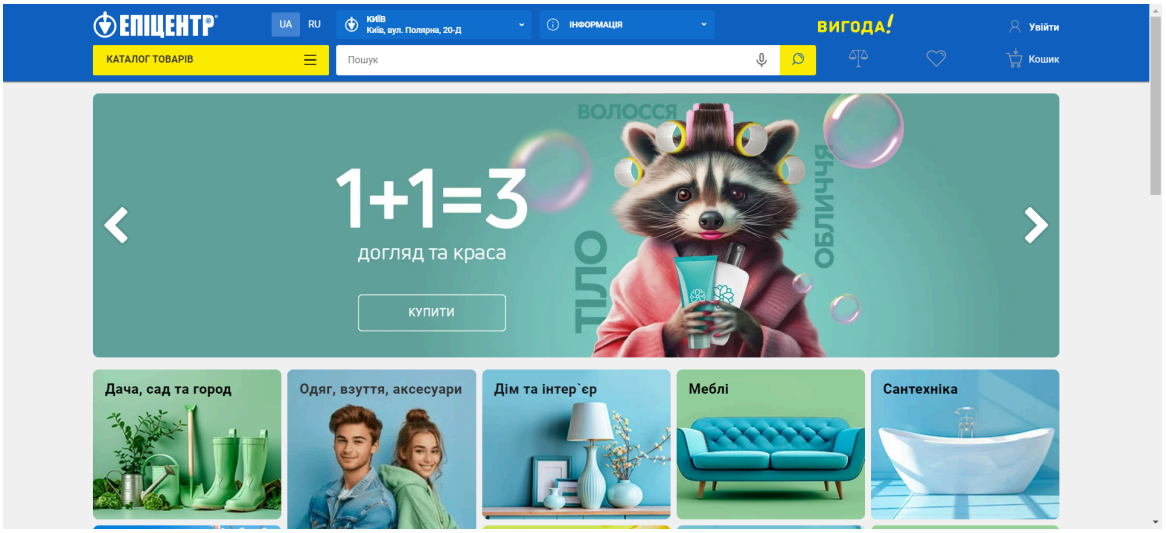


рисунок 1.3 – Головна сторінка сайту epicentrk.ua

Переваги:

– широкий асортимент товарів: Епіцентр пропонує широкий асортимент будівельних матеріалів, інструментів, товарів для дому та саду;

– доступні ціни: Епіцентр часто пропонує конкурентні ціни на будівельні матеріали, а також акції та розпродажі;

– розвинена мережа магазинів: Епіцентр має розгалужену мережу магазинів по всій Україні, що робить його доступним для більшості покупців;

– можливість замовлення онлайн з доставкою або самовивозом: Епіцентр пропонує можливість замовлення онлайн з доставкою або

самовивозом з магазину;

– професійна консультація менеджерів: Епіцентр пропонує професійну консультацію менеджерів щодо будівельних матеріалів.

Недоліки:

– не завжди актуальна інформація про наявність товару на сайті: Інформація про наявність товару на сайті Епіцентр не завжди актуальна, що може призвести до незручностей для покупців;

– скарги на затримки з доставкою: Деякі покупці скаржаться на затримки з доставкою товарів, замовлених онлайн;

– не завжди якісне обслуговування в магазинах: Деякі покупці скаржаться на неякісне обслуговування в магазинах Епіцентр;

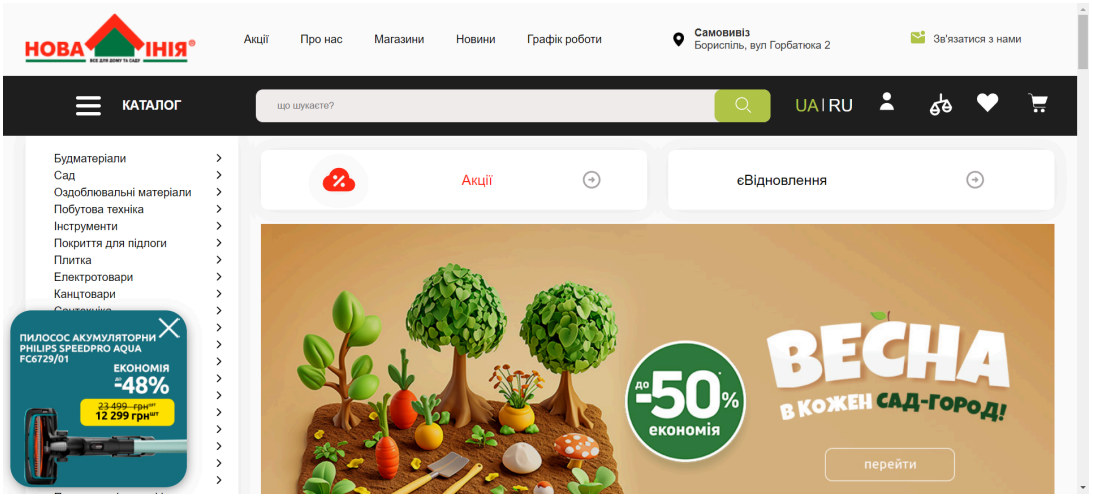


Рисунок 1.4 – Головна сторінка сайту nl.ua

Переваги:

– великий вибір будівельних матеріалів та інструментів: Нова Лінія пропонує широкий вибір будівельних матеріалів та інструментів;

– доступні ціни: Нова Лінія часто пропонує конкурентні ціни на будівельні матеріали, а також акції та розпродажі;

– зручний сайт: Сайт Нова Лінія зручний для навігації та пошуку потрібних товарів;

– швидка доставка: Нова Лінія пропонує швидку доставку по всій

Україні.

Недоліки:

– невелика мережа магазинів: Нова Лінія має меншу мережу магазинів, ніж Епіцентр, що може бути менш зручним для деяких покупців;

– не завжди є в наявності всі товари, представлені на сайті: Інформація про наявність товару на сайті Нова Лінія не завжди актуальна, що може призвести до незручностей для покупців;

– скарги на неякісне обслуговування: Деякі покупці скаржаться на неякісне обслуговування в магазинах Нова Лінія.

Висновки до розділу 1

У розділі було проведено дослідження інтернет магазинів з продажу будівельних матеріалів.

Основними складовими бізнесу продажу будівельних матеріалів, як показало дослідження, є:

– широкий асортимент;

– низькі ціни;

– часті знижки та промо акції;

– хороші консультанти;

– наявність швидкої та надійної доставки.

Досліджуючи сайти конкурентів, було виявлено ряд спільних рис, серед яких:

– структурована та логічна подача інформації;

– привабливий та інтуїтивний дизайн;

– адаптивність;

– строгість в стилі.

РОЗДІЛ 2. ПРОЕКТУВАННЯ САЙТУ

2.1 Загальна характеристика досліджуваної проблеми

У сучасних умовах стрімкого розвитку інформаційних технологій веб-розробка займає важливе місце у всіх сферах діяльності, зокрема у торгівлі. Онлайн-присутність стає критично важливою для успішного ведення бізнесу, адже інтернет-магазини пропонують значно більше можливостей для залучення клієнтів порівняно з традиційними оффлайн-магазинами. Під час розробки інтернет-магазину для продажу будівельних матеріалів, який є об'єктом даного дослідження, виникає необхідність у виборі оптимальних технологій та інструментів для створення функціонального і зручного для користувача веб-сайту.

Інтернет-магазини надають можливість значно розширити географію бізнесу, адже через інтернет можна продавати товари не лише у межах одного міста чи регіону, а й по всій країні або навіть світу. Це особливо актуально для будівельних матеріалів, які можуть бути необхідні в різних регіонах. Однак, розробка інтернет-магазину пов'язана з рядом викликів, таких як забезпечення зручності користування, швидке завантаження сторінок, ефективний пошук товарів, інтеграція з платіжними системами, забезпечення безпеки особистих даних користувачів та інше.

Розробка інтернет-магазину для будівельних матеріалів має свої особливості, зокрема необхідність надання детальної інформації про кожен товар, включаючи технічні характеристики, фото, відеоогляди, відгуки користувачів тощо.

Це вимагає продуманої структури сайту, ефективної системи управління контентом та оптимізації для пошукових систем.

2.2 Обґрунтування вибору мови та технології програмування

Для реалізації проекту було обрано такі мови програмування та технології:

HTML (HyperText Markup Language) - мова розмітки для створення веб-сторінок та їх структуризації. HTML використовується для побудови каркасу веб-сторінки, забезпечуючи логічну структуру контенту. Це базова мова, яка підтримується всіма браузерами та є основою веб-розробки.

CSS (Cascading Style Sheets) - мова стилізації для опису зовнішнього вигляду веб-сторінок. CSS дозволяє відокремити зміст від презентації, забезпечуючи можливість змінювати стиль сайту без змін його структури. Використання CSS дозволяє зробити сайт привабливим і зручним для користувача.

JavaScript - мова програмування для створення інтерактивних елементів на веб-сторінках. JavaScript дозволяє реалізувати динамічну взаємодію з користувачем, забезпечуючи зворотній зв'язок в режимі реального часу. Завдяки JavaScript можна створювати складні інтерактивні елементи, такі як слайдери, спливаючі вікна, валідація форм тощо.

Python - високорівнева мова програмування, що забезпечує простоту написання та читання коду. Python відомий своєю зручністю і універсальністю, що робить його ідеальним вибором для веб-розробки, зокрема при використанні фреймворку Django.

Для прискорення процесу розробки та забезпечення високої функціональності веб-сайту були використані наступні фреймворки:

Django - високорівневий веб-фреймворк для Python, що дозволяє швидко створювати складні та масштабовані веб-застосунки. Django забезпечує зручну роботу з базами даних, а також має вбудовані механізми для авторизації та аутентифікації користувачів. Це дозволяє розробникам зосередитися на функціональності додатку, а не на написанні рутинного коду. Django також забезпечує високий рівень безпеки завдяки вбудованим захисним механізмом

від поширених атак, таких як SQL-ін'єкції та XSS.

TailwindCSS - утилітарний CSS-фреймворк, що надає набір готових класів для швидкого створення адаптивного дизайну. Tailwind CSS дозволяє розробникам створювати стильові рішення без написання додаткового CSS-коду, що значно пришвидшує процес розробки. Використання Tailwind забезпечує послідовність у стилях і дозволяє легко змінювати дизайн на різних етапах проекту.

Проектування веб-сайту інтернет-магазину почалося з розробки його прототипу. Було визначено основні функціональні вимоги:

Зручний та інтуїтивно зрозумілий інтерфейс для користувачів. Це означає, що користувачі повинні мати можливість легко знаходити потрібні їм товари, здійснювати пошук за ключовими словами, а також фільтрувати товари за різними параметрами (наприклад, за ціною, брендом, характеристиками).

Можливість перегляду каталогу товарів та сортування за різними параметрами. Каталог товарів повинен бути структурований таким чином, щоб користувачі могли легко орієнтуватися в асортименті продукції. Це може включати поділ на категорії, підкатегорії та спеціальні розділи (наприклад, новинки, розпродажі).

Особистий кабінет користувача, що дозволяє переглядати історію покупок та керувати налаштуваннями облікового запису. Особистий кабінет повинен бути зручним і безпечним, забезпечуючи користувачам доступ до їх персональної інформації та можливість змінювати налаштування профілю.

Кошик для покупок з функцією швидкого оформлення замовлення. Кошик повинен бути інтегрований з системою управління замовленнями, забезпечуючи можливість швидкого та зручного оформлення покупок.

Оптимізація веб-сайту була здійснена шляхом використання сучасних інформаційних технологій та підходів, таких як:

Використання AJAX для асинхронного завантаження даних, що дозволяє покращити взаємодію з користувачем без необхідності перезавантаження сторінки. Це забезпечує більш швидке та зручне користування сайтом, оскільки

користувачі можуть отримувати оновлену інформацію в режимі реального часу.

Застосування методів пошукової оптимізації (SEO) для покращення видимості сайту в пошукових системах. Оптимізація контенту, структури сайту та використання ключових слів дозволяє покращити рейтинг сайту у пошукових системах, що сприяє збільшенню кількості відвідувачів.

Адаптивний дизайн, що забезпечує коректне відображення сайту на різних пристроях, включаючи мобільні телефони та планшети. Це означає, що веб-сайт буде виглядати та функціонувати однаково добре на всіх пристроях, що є важливим фактором для залучення та утримання користувачів.

Безпека веб-сайту є критично важливою складовою, тому були обрані наступні заходи:

Використання HTTPS для шифрування даних між клієнтом та сервером. Це забезпечує захист переданих даних від несанкціонованого доступу та перехоплення, що є особливо важливим при передачі особистих даних користувачів та інформації про платіжні операції.

Впровадження механізмів захисту від CSRF (Cross-Site Request Forgery) та XSS (Cross-Site Scripting) атак. Це забезпечує захист від небажаних дій, які можуть бути здійснені зловмисниками від імені користувача, а також від впровадження шкідливого коду на веб-сторінках.

Регулярні оновлення програмного забезпечення для усунення вразливостей. Це включає оновлення серверного програмного забезпечення, баз даних, бібліотек та фреймворків, які використовуються у проекті, що дозволяє підтримувати високий рівень безпеки.

2.2 Структура сайту

2.2.1 Загальна структура сайту

Після того як було визначено технології за допомогою яких буде реалізований проект потрібно створити структуру сторінок веб сайту (рис. 2.1) .

Ці сторінки є повинні реалізовувати увесь функціонал сайту та відповідати вимогам користувача. В моєму сайті є ось такі сторінки :

– головна сторінка;
– каталог;
– про нас;
– профіль користувача;
– кошик;
– логін;
– реєстрація;
– сторінка редагування даних користувача;
– сторінка товару;
– скидання паролю;
– сторінка оформлення замовлення;
– сторінка даних про оформленне замовлення.

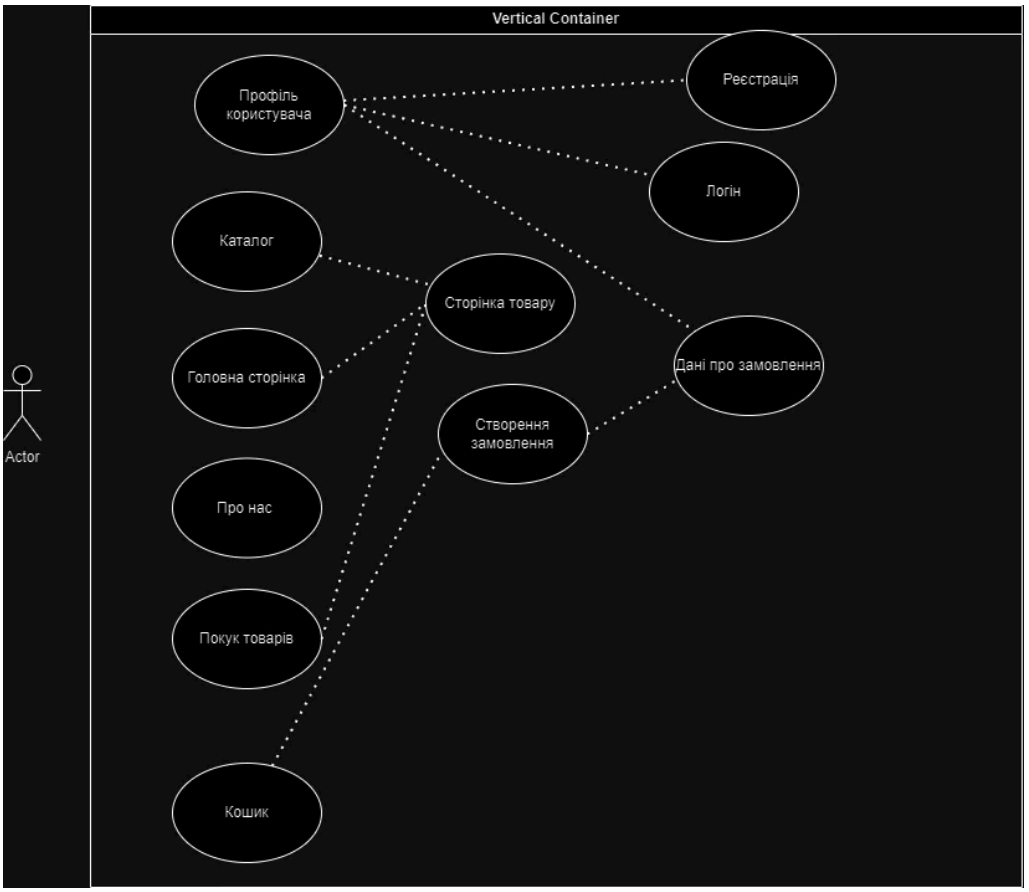


Рисунок 2.1 – UML діаграма сайту

2.2.2 Навігаційна панель

Навігація по сайту здійснюється за допомогою навігаційної панелі яка знаходиться в хедері сторінок.

Кнопки пересилають користувача на сторінки :

– головна;

– каталог;

– про нас;

– профіль користувача;

– кошик.

Також навігаційна панель містить поле пошуку .

Рисунок 2.2 – Навігаційна панель сайту

2.2.3 Головна сторінка

Відкриваючи сайт користувач бачить головну сторінку сайту (рис 2.3) .

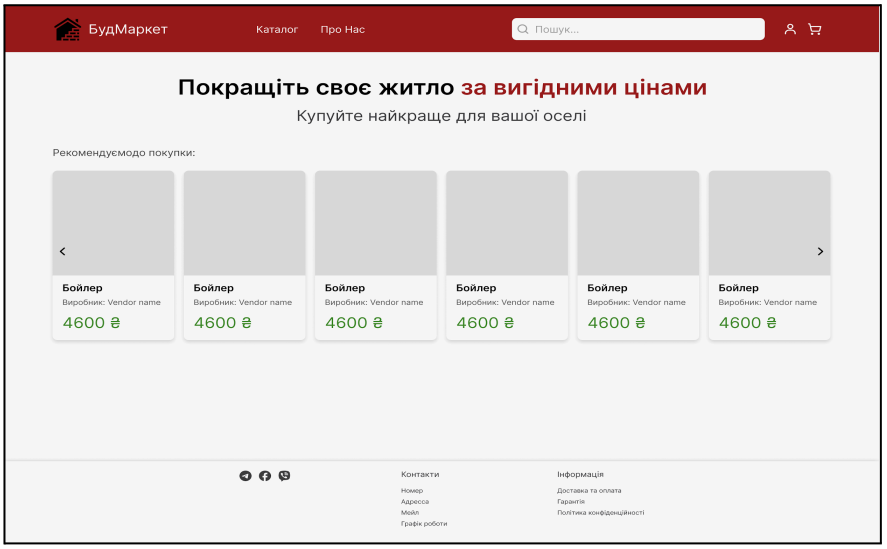


Рисунок 2.3 – Wireframe головної сторінки сайту

Основний контент включає рекламний заголовок, секцію рекомендованих товарів у вигляді каруселі, де відображаються товарні картки з зображеннями, назвами, виробниками та цінами. У футері розміщені контактні дані та інформаційні посилання, а також іконки соціальних мереж. Сторінка забезпечує навігацію, рекламу та зручність для користувачів, спрямовуючи їх до здійснення покупок.

2.2.4 Каталог

Сторінка каталогу (рис. 2.4) містить верхнє меню з посиланнями на розділи, поле пошуку, іконки для кошика та користувача. Основний контент включає заголовок "Каталог" і бічну панель з фільтрами для сортування товарів за різними критеріями, вибору категорій і брендів.

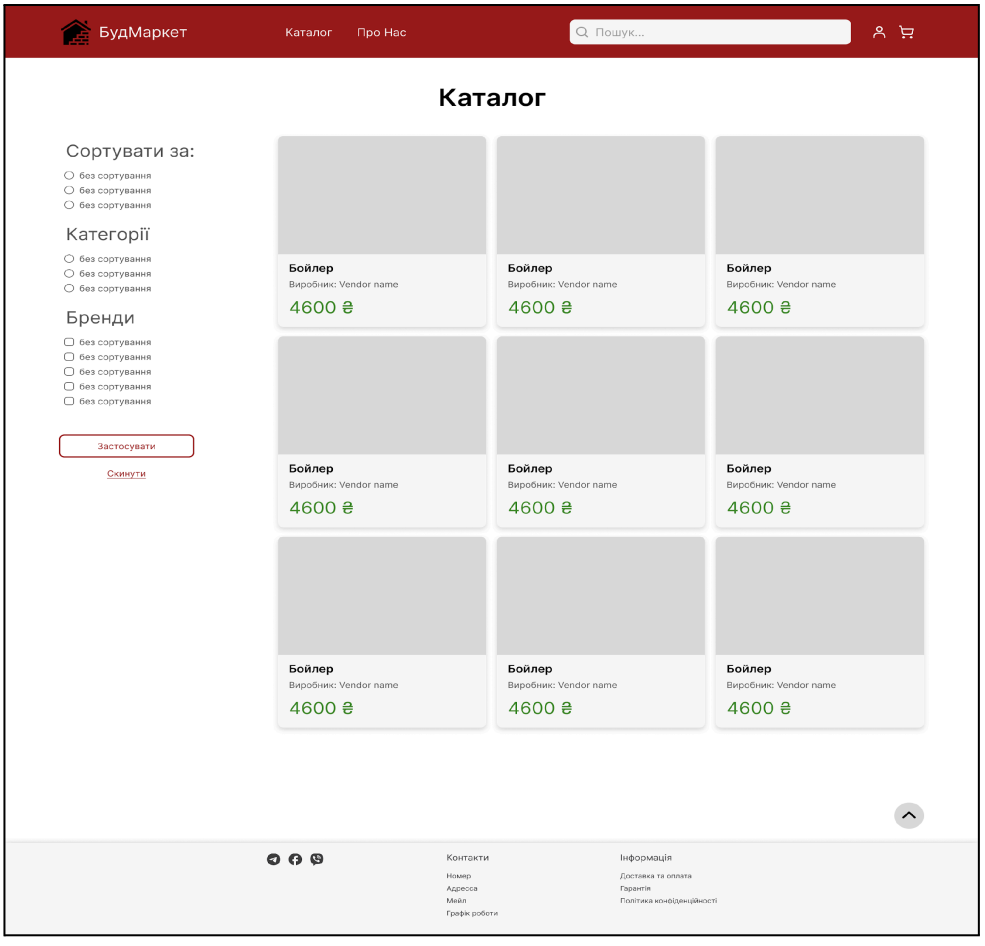


Рисунок 2.4 – Wireframe сторінки каталогу

Товари представлені у вигляді сітки з картками, які містять зображення, назву, виробника та ціну. У футері розміщені контактні дані, інформація про доставку, оплату, гарантії, політику конфіденційності та іконки соціальних мереж.

Ця сторінка забезпечує зручний доступ до товарів магазину, допомагаючи користувачам швидко знаходити потрібні товари за допомогою фільтрів та сортування. Вона також надає основну інформацію про товари, сприяючи прийняттю рішення про покупку.

Інтуїтивно зрозумілий інтерфейс та зручне розміщення елементів покращують користувацький досвід.

2.2.5 Про нас

Сторінка "Про нас" інтернет-магазину "БудМаркет" містить верхнє меню з посиланнями на розділи "Каталог" та "Про нас", поле пошуку, іконки для кошика та користувача.

Основний контент включає заголовок "Про нас" та текст, який описує діяльність магазину, що працює більше 10 років та пропонує широкий асортимент будівельних та господарських товарів.

У тексті підкреслюється високий рівень якості продукції та відмінне обслуговування клієнтів. Контактні дані, включаючи номер телефону, електронну пошту та посилання на соціальні мережі, розміщені поруч із картою, що вказує на місце розташування магазину. Також надано графік роботи.

Футер сторінки містить додаткові розділи з контактною інформацією, адресою, даними про доставку та оплату, гарантії, політику конфіденційності, а також іконки соціальних мереж.

Ця сторінка надає детальну інформацію про магазин, сприяє встановленню довіри до нього та забезпечує легкий доступ до контактних даних, що полегшує спілкування з магазином та отримання консультацій.

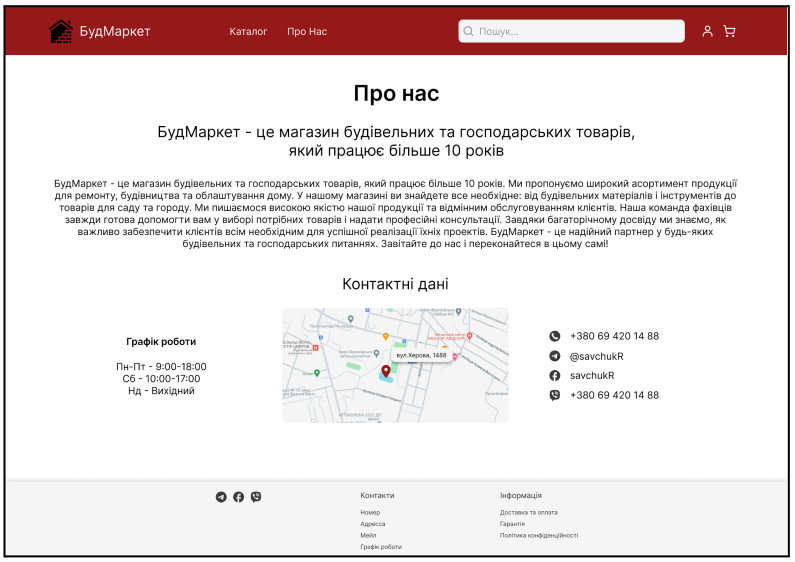


Рисунок 2.5 – Wireframe сторінки Про Нас

2.2.6 Профіль користувача

Сторінка "Профіль користувача" (рис. 2.6) інтернет-магазину "БудМаркет" містить верхнє меню з посиланнями на розділи, поле пошуку, іконки для кошика та користувача.

Основний контент включає заголовок "Логін" та персональну інформацію користувача. Відображаються ім'я та прізвище, електронна пошта, телефон і адреса доставки. Кнопка редагувати переносить на сторінку редагування даних користувача що може бути корисним в випадку заміни номеру телефону або електронної пошти .

Нижче розташований розділ "Мої замовлення", який розділений на дві частини: "Активні" та "Неактивні" замовлення. Активні замовлення містять поточні та відправлені замовлення, а неактивні включають виконані та скасовані замовлення. Кожне замовлення відображається у вигляді картки з номером замовлення та його статусом.

Ця сторінка забезпечує користувачів легким доступом до їх персональної інформації та історії замовлень, допомагаючи швидко переглядати та керувати активними і неактивними замовленнями. Інтуїтивно зрозумілий інтерфейс та зручне розміщення елементів покращують користувацький досвід.

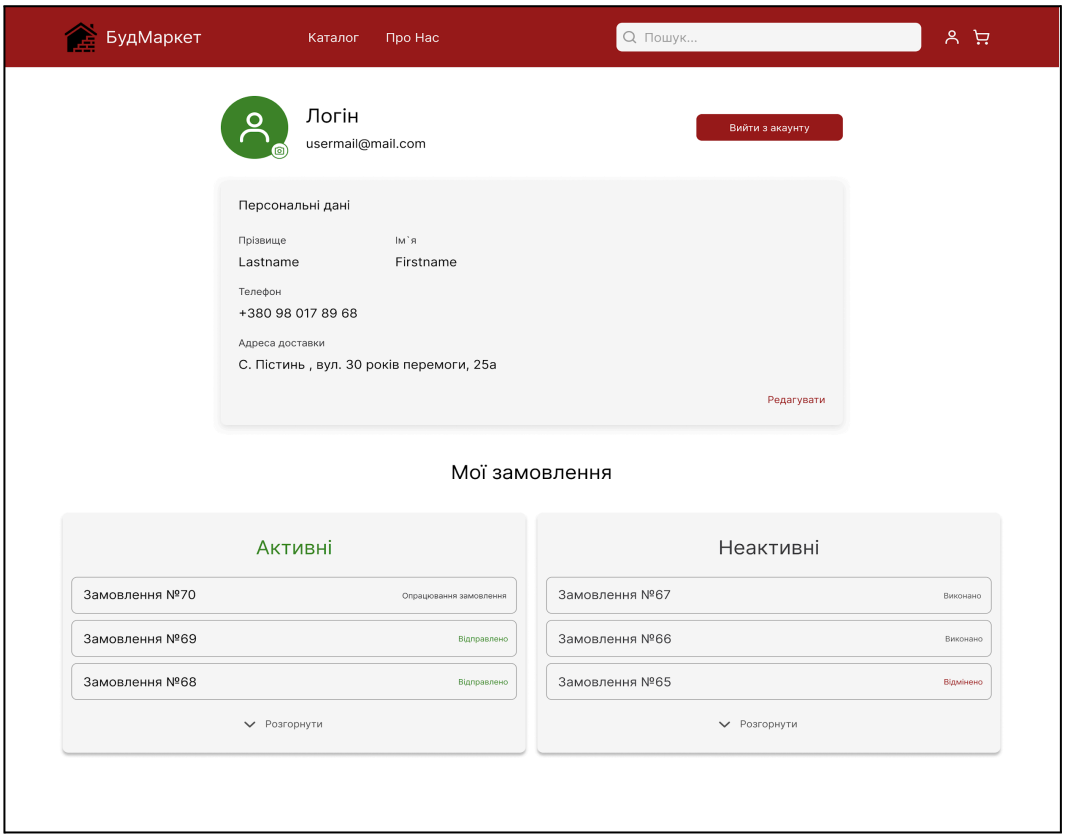


Рисунок 2.6 – Wireframe сторінки профілю користувача

2.2.7 Кошик

Сторінка "Кошик замовлень" (рис. 2.7) інтернет-магазину "БудМаркет" містить верхнє меню з посиланнями на розділи "Каталог" та "Про Нас", поле пошуку, іконки для кошика та користувача. Основний контент включає заголовок "Кошик замовлень" та список товарів, що додані до кошика.

Кожен товар відображається у вигляді картки з назвою, кількістю в наявності, ціною та сумою. Користувач може змінювати кількість товарів за допомогою кнопок "+" і "–". Крім того, є кнопка для видалення товару з кошика.

Внизу сторінки розташована загальна сума замовлення, а також кнопки "Продовжити покупки" та "Оформити замовлення". Інтерфейс інтуїтивно зрозумілий і забезпечує зручне керування замовленням та швидкий перехід до оформлення покупки.

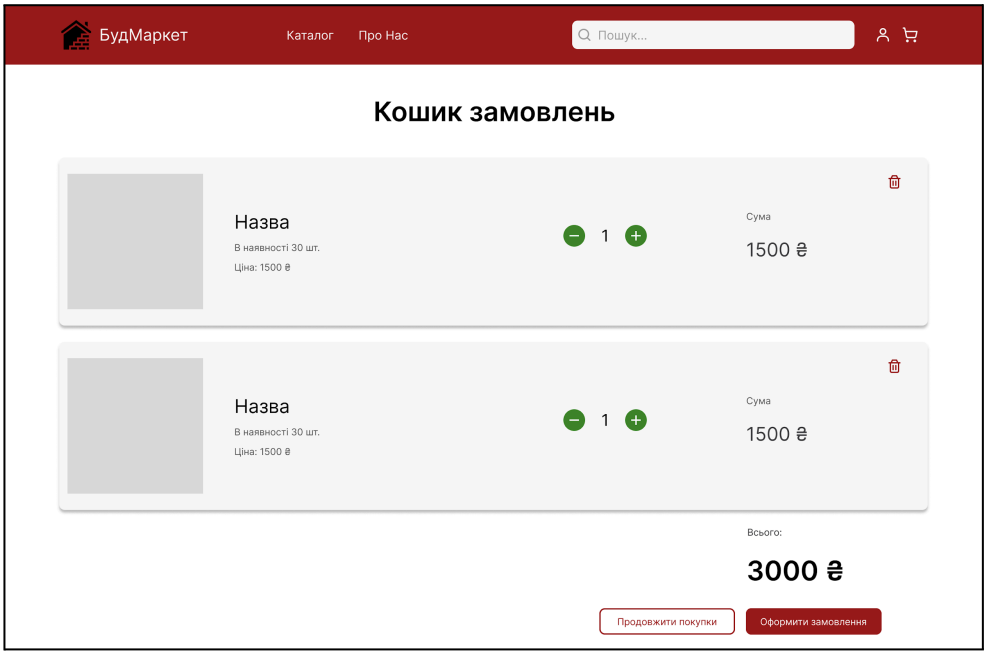


Рисунок 2.7 – Wireframe сторінки кошику

2.2.8 Логін та Реєстрація

Сторінка "Реєстрація" містить основний контент включає заголовок "Реєстрація" та форму для створення нового облікового запису.

Форма містить поля для введення логіну, електронної пошти, пароля та підтвердження пароля. Користувачі можуть заповнити ці поля для реєстрації на сайті. Під формою розташована зелена кнопка "Зареєструватися", яка завершує процес реєстрації.

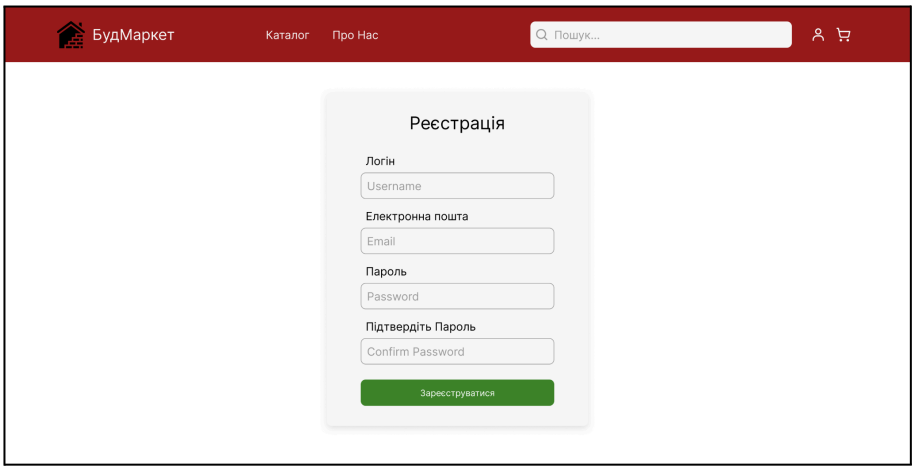


Рисунок 2.8 Wireframe сторінки реєстрації

Сторінка "Логін" інтернет-магазину "БудМаркет" містить верхнє меню з посиланнями на розділи, поле пошуку, іконки для кошика та користувача. Основний контент включає заголовок "Логін" та форму для входу до облікового запису.

Форма для входу містить поля для введення логіну та пароля. Під формою розташована кнопка "Увійти", яка дозволяє користувачам увійти в свій обліковий запис.

Ця сторінка забезпечує користувачам легкий доступ до їх облікового запису, допомагаючи швидко авторизуватися та отримати доступ до персоналізованих функцій сайту. Інтуїтивно зрозумілий інтерфейс та зручне розміщення елементів покращують користувацький досвід.

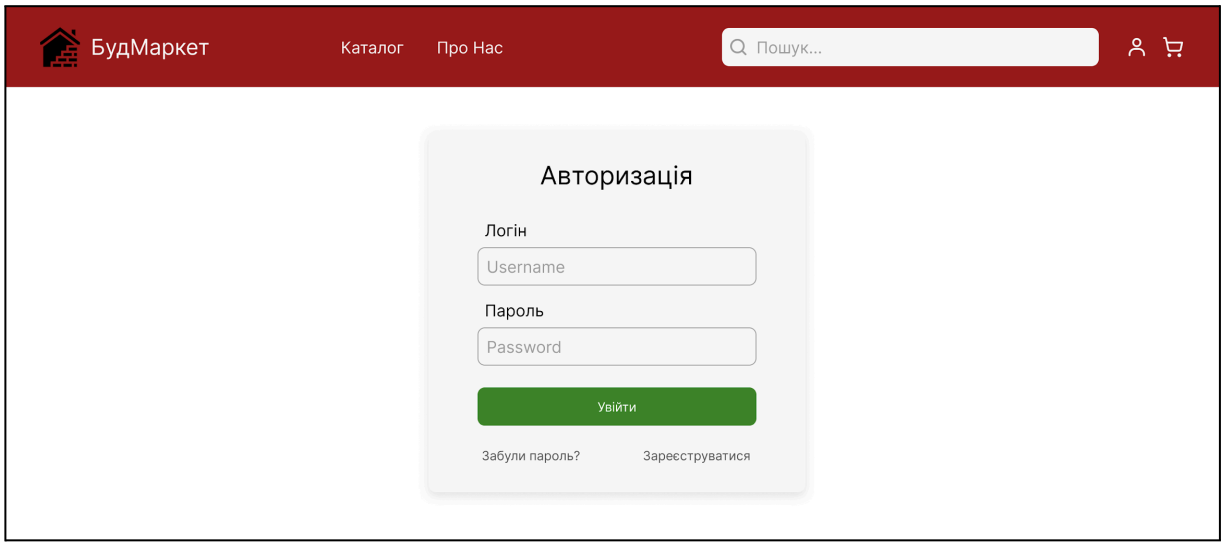


Рисунок 2.9 – Wireframe сторінки логіну

2.2.9 Редагування даних користувача

За допомогою сторінки редагування даних користувач може змінити дані профілю. Ця сторінка необхідна якщо користувачу потрібно оновити дані профілю, змінити адрес електронної пошти чи пароль .

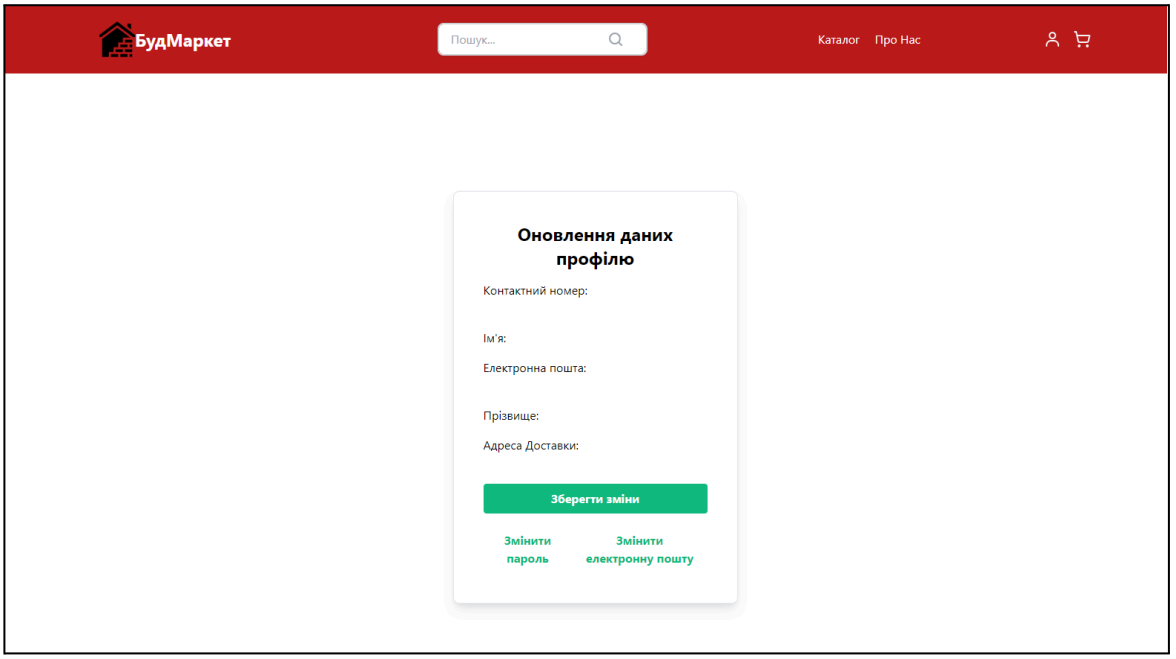


Рисунок 2.10 – Скріншот сторінки редагування даних користувача

2.2.10 Деталі товару

Сторінка деталей товару (рис. 2.11) дозволяє користувачам дізнатися необхідну інформацію про товар на лівій стороні сторінки розташоване велике зображення товару з можливістю перегортання. Праворуч знаходиться блок інформації про товар, який включає назву товару великим шрифтом, ім'я виробника, опис товару та основні характеристики (характеристика 1, характеристика 2, характеристика 3).

Під інформаційним блоком вказана ціна товару в гривнях. Нижче розташована панель вибору кількості товару з кнопками "+" і "–" для зміни кількості. Поруч є червона кнопка "Buy now" для негайної покупки та кнопка з іконкою кошика для додавання товару до кошика.

Ця сторінка забезпечує користувачів всією необхідною інформацією про товар та дозволяє легко здійснити покупку або додати товар до кошика для подальшого придбання. Інтуїтивно зрозумілий інтерфейс та зручне розміщення елементів покращують користувацький досвід.

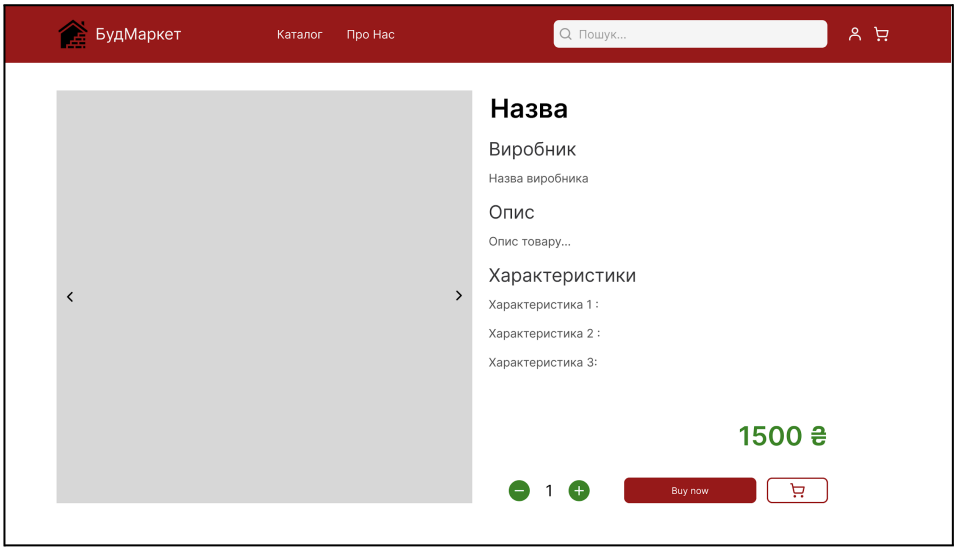


Рисунок 2.11 – Wireframe сторінки деталей про товар

2.2.11 Скидання паролю

Сторінка "Скидання паролю" інтернет-магазину "БудМаркет" містить форму для введення скидання паролю. Форма складається з одного поля вводу для email та кнопки "Скинути" зеленого кольору.

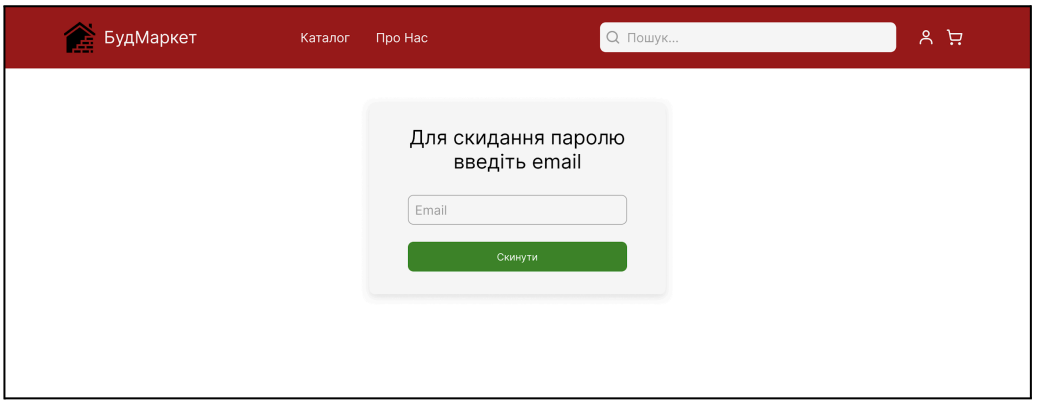


Рисунок 2.12 – Wireframe сторінки скидання паролю

2.2.12 Оформлення замовлення

Сторінка "Оформлення замовлення" (рис. 2.13) інтернет-магазину "БудМаркет" містить верхнє меню з посиланнями на розділи, поле пошуку,

іконки для кошика та користувача. Основний контент включає заголовок "Оформлення замовлення" та форму для заповнення інформації про замовлення. Форма розділена на три частини:

– Контактні дані:
– поле для введення номера телефону;
– поле для введення імені;
– поле для введення прізвища.

Доставка:

– вибір між доставкою та самовивозом;
– поле для введення адреси доставки у разі вибору доставки.

Оплата:

– Вибір способу оплати (готівка або карта)

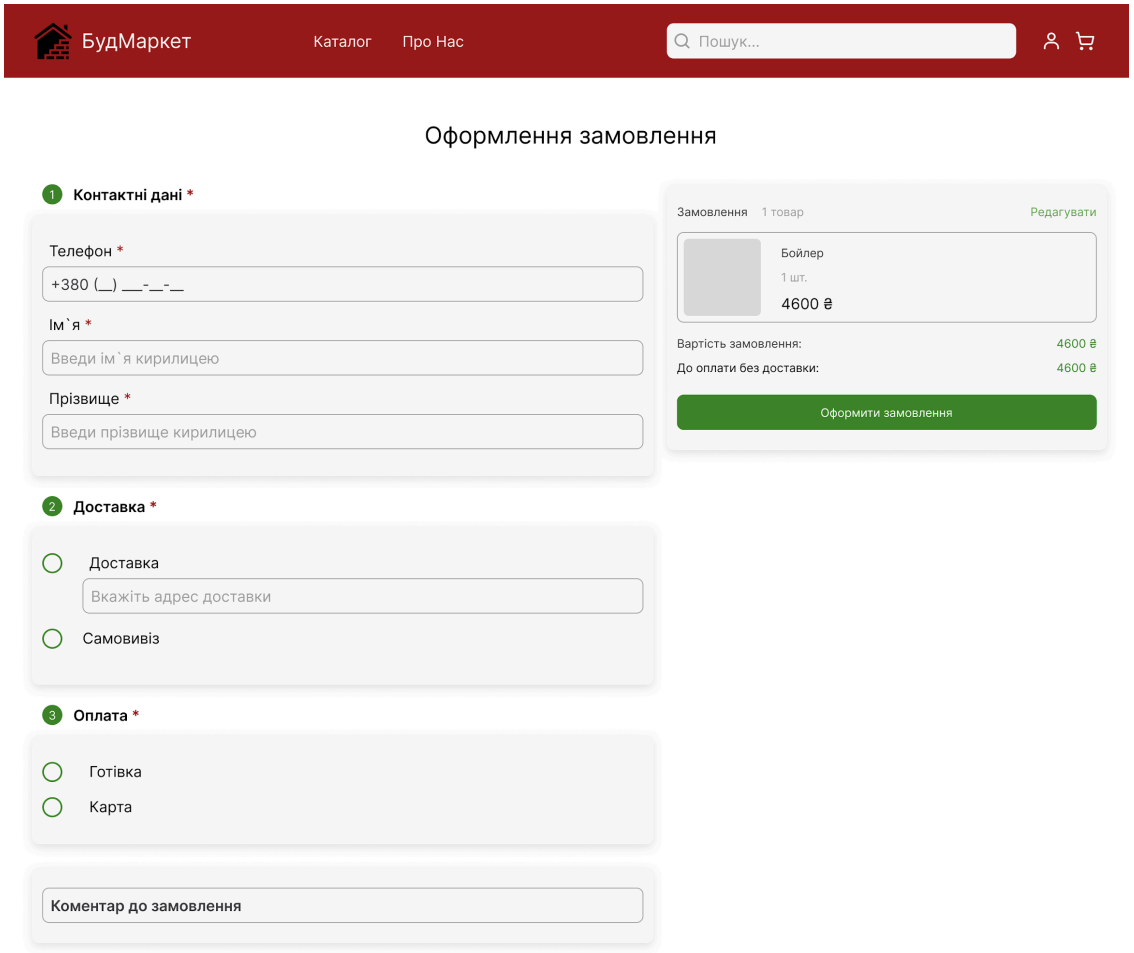


Рисунок 2.13 – Wireframe сторінки оформлення замовлення

Нижче розташоване поле для коментаря до замовлення. Праворуч від форми відображається підсумок замовлення з детальною інформацією про товар, його кількість та загальну вартість. Підсумок включає кнопку "Оформити замовлення" для завершення процесу покупки.

Ця сторінка забезпечує користувачам легкий доступ до завершення покупки, допомагаючи швидко і зручно оформити замовлення та вибрати спосіб доставки і оплати. Інтуїтивно зрозумілий інтерфейс та зручне розміщення елементів покращують користувацький досвід.

2.2.13 Дані про замовлення

Сторінка "Підтвердження замовлення" інтернет-магазину "БудМаркет" містить верхнє меню з посиланнями на розділи, поле пошуку, іконки для кошика та користувача. Основний контент включає заголовок "Дякуємо за ваше замовлення!" та деталі замовлення. Основна частина сторінки складається з двох розділів:

Деталі замовлення:

1. Номер замовлення;
2. Список товарів із зазначенням кількості та ціни кожного товару;
3. Загальна сума замовлення;

Контактні дані:

1. Ім’я та прізвище покупця;
2. Номер телефону;
3. Спосіб доставки;
4. Спосіб оплати;
5. Коментар до замовлення;

Нижче розташовані кнопки "Продовжити покупки" та "Переглянути замовлення", що дозволяють користувачу легко повернутися до покупок або переглянути деталі зробленого замовлення.

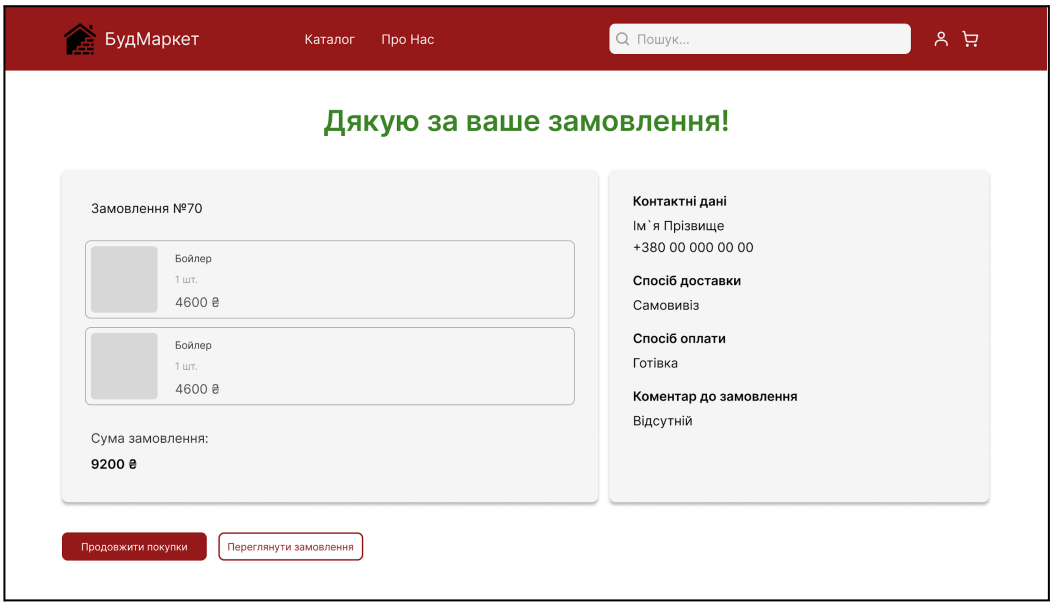


Рисунок 2.14 – Wireframe сторінки даних про замовлення

Ця сторінка забезпечує користувачів всією необхідною інформацією про підтверджене замовлення, допомагаючи їм швидко переглянути та перевірити деталі покупки. Інтуїтивно зрозумілий інтерфейс та зручне розміщення елементів покращують користувацький досвід.

Висновок до розділу 2

У другому розділі кваліфікаційної роботи було проведено детальне проектування веб-сайту інтернет-магазину будівельних матеріалів. Основні акценти зроблені на створення зручного для користувача інтерфейсу, структурованої архітектури сайту та інтеграції необхідних функціональних можливостей. Основні результати другого розділу включають:

Аналіз потреб користувачів та функціональних вимог: Було проведено детальне дослідження вимог, що висуваються до сучасних інтернет-магазинів, включаючи зручність навігації, швидкість завантаження сторінок, та доступність інформації про товари.

На основі аналізу було створено схеми навігації та структури сторінок, що забезпечують логічну та інтуїтивно зрозумілу подачу інформації. Основними

елементами сайту стали головна сторінка, каталог товарів, сторінка товару, кошик, та особистий кабінет користувача.

Було розроблено макети сторінок з урахуванням вимог сучасного веб-дизайну та адаптивності. Особлива увага приділялася зручності користування та естетичному вигляду сайту.

Визначено та спроектовано основні функціональні модулі сайту, такі як система пошуку та фільтрації товарів, механізм додавання товарів у кошик, оформлення замовлення, та система управління користувачами.

РОЗДІЛ 3. РОЗРОБКА САЙТУ

3.1 Функціонал сайту

Для розробки сайту був обраний фреймворк Django для мови програмування Python. Це зручний інструмент для розробки сайту який має багато функцій які спрощують розробку та підтримку сайту .Також для створення зовнішнього вигляду сторінок крім стандартних HTML та CSS я використовував фреймворк TailwindCSS .

3.2 Додатки

В Django різноманітні функції сайтів реалізується за допомогою додатків . Це дозволяє окремо розробляти та додавати на сайт різноманітні функції не заважаючи працювати іншим. Тому в моєму проекті сайт був реалізований за допомогою 4 додатків:

– Market це додаток що відповідає за відображення основних сторінок сайту, функціонал додавання та редагування даних про товар , функціонал каталогу.

– Users це додаток що відповідає за реєстрацію, авторизацію та роботою з акаунтами користувачів.

– Cart це додаток що відповідає за функціонал кошика

– Order це додаток що відповідає за роботу з створення замовлень , відстежування статусу замовлення , інформацію про дані замовлення.

В середині кожного додатку містяться файли основними є views.py , models.py , urls.py, forms.py . Вони відповідають за роботу додатку та функціоналу сторінок.

Додатки можуть обмінюватися даними посилаючись один на одний тому щоб детально пояснити ці зв'язки я розгляну кожний додаток .

3.3 Додаток Market

Додаток Market це основний додаток проекту оскільки він відповідає за відображення таких сторінок як :

3.3.1 Базовий шаблон

Базовий шаблон в Django відіграє ключову роль у забезпеченні консистентності, простоти та ефективності розробки веб-додатків.

Базовий шаблон містить загальне оформлення сайту, включаючи заголовок (<head>), навігацію, підвал (<footer>), стилі та скрипти. Це забезпечує єдиний зовнішній вигляд на всіх сторінках сайту.

За допомогою базового шаблону можна легко підтримувати і змінювати дизайн всього сайту, вносячи зміни лише в одному місці.

Елементи, які повторюються на багатьох сторінках, такі як навігаційна панель, логотип, футер і інші, визначаються один раз у базовому шаблоні.

Це забезпечує консистентність та знижує ймовірність помилок, оскільки немає потреби дублювати код на кожній сторінці.

Використання блоків ({% block content %}) дозволяє визначити місця, де буде відображатися контент кожної сторінки вміст.

Це дозволяє створювати нові сторінки швидше та зручніше, просто наслідуючи базовий шаблон і додаючи необхідний контент в визначені блоки.

Зміни, що стосуються загальної структури або оформлення сайту, потрібно вносити лише в базовий шаблон, що значно спрощує підтримку та оновлення сайту.

Наприклад, зміна кольору фону або шрифта буде автоматично застосована до всіх сторінок, які наслідують базовий шаблон.

```
<!DOCTYPE html>
{% load static %}
<html lang="ua">
```

```
    <head>
        <meta charset="UTF-8" name="viewport"
content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" href="{% static 'market/styles.css'
%}">
        <link
href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min
.css" rel="stylesheet">
    </head>
```

Першим в цьому шаблоні оголошується тип документу HTML 5. Далі завантажується статична файлова система Django і визначається мова документу : Українська. В тезі head відбувається :

- Налаштування кодування символів UTF-8 та адаптивного дизайну.
- Підключення власного стилю styles.css через {% static 'market/styles.css' %}.
- Підключення Tailwind CSS для стилізації через CDN.

Основний вміст шаблону знаходиться всередині тегу body . Саме в ньому я створив навігаційну панель сайту так званий header який завжди знаходиться над іншими частинами сайту та футор який містить додаткову інформацію та завжди знаходиться знизу всіх елементів сторінки . Код в додатку А.

Як раніше було сказано для створення зовнішнього вигляду сайту та роботи зі стилями використовується TailwindCSS за допомогою якого визначаються :

- розмір елементів сторінки;
- відступи між елементами;
- розташування елементів на екрані;
- розмір , шрифт та колір тексту ;
- колір заднього фону.

Для реалізації пошуку товарів використовується форма з текстовим полем та кнопкою для відправлення запиту. Форма надсилає запит методом GET на сторінку каталогу. Нижче знаходяться кнопки які переносять користувача на

сторінки каталогу та “Про Нас”. Код в додатку А.

Далі містяться кнопки з піктограмами які переносять користувача на сторінки кошика та профілю користувача. Для того щоб піктограми коректно відображалися на різних пристроях з різною роздільною здатністю використовується формат SVG.

```
{% block content %}

{% endblock content %}
```

В середині блоку контент буде відображатися вміст сторінок які наслідуватимуть даний шаблон.

В додатку А знаходиться код футору в якому знаходяться піктограми соціальних мереж які містять посилання на соціальні мережі магазину та список посилань на додаткову інформацію про магазин .

3.3.2 Головна

Код сторінки починається з оголошення типу документу та наслідування базового шаблону. Встановлюється фон сторінки світло-сірого кольору та основний контейнер з відступами і центруванням для вмісту.

В центрі екрану створюється заголовок та підзаголовок. Нижче відображається полоса прокрутки з рекомендованими товарами та мінімальною інформацією про них.

Скрипт для прокрутки каруселі продуктів. Визначає кількість прокрутки (scrollAmount) та обробники подій для кнопок прокрутки.

```
<script>
        const scrollContainer =
document.getElementById('products-carousel');
        const scrollAmount = scrollContainer.clientWidth;
```

```
            document.getElementById('scroll-left').onclick = function
() {
                scrollContainer.scrollBy({
                    left: -scrollAmount,
                    behavior: 'smooth'
                });
            };

            document.getElementById('scroll-right').onclick = function
() {
                scrollContainer.scrollBy({
                    left: scrollAmount,
                    behavior: 'smooth'
                });
            };
        </script>
        </body>
    {% endblock content %}
```

3.3.3 Про нас

Сторінка Про Нас є доволі статичною що означає що вона не змінює своє наповнення відповідно до товарів на сайті оскільки їй це не потрібно. В ній міститься опис магазину та інформація про нього. В ній відображаються контакти як зв'язатися з магазином , інформація про його роботу та інтегрована карта на якій можна знайти його розташування. Код сторінки в додатку А.

3.3.4 Каталог

Бічна панель з фільтрами:

– форма сортування: користувачі можуть сортувати продукти за мінімальною або максимальною ціною, або не застосовувати сортування;

– категорії: вибір категорій дозволяє користувачам відфільтрувати продукти за певною категорією;

– бренди: можна вибрати один або кілька брендів для відображення продуктів тільки вибраних брендів.

Головний екран:

1. Пошук: форма пошуку дозволяє користувачам здійснювати пошук продуктів за ключовими словами.

2. Сітка продуктів: продукти відображаються у вигляді карток у сітці. Кожна картка містить зображення продукту, назву, бренд та ціну.

3. Пагінація: забезпечує навігацію між сторінками продуктів, відображаючи посилання на попередню та наступну сторінки, а також на першу та останню сторінки.

Функціонал сторінки :

1. Сортування та фільтри: реалізовані за допомогою форм, які передають параметри сортування та фільтрування через метод GET.

2. Відображення продуктів: продукти отримуються через контекст, наданий шаблону, і відображаються в сітці з використанням циклу for.

3. Скидання фільтрів: наявна можливість скидання всіх фільтрів та сортувань.

Цей шаблон забезпечує зручний та інтуїтивний інтерфейс для користувачів, дозволяючи їм легко знаходити та переглядати продукти за їхніми вподобаннями.

3.3.5 Деталі товару

Сторінка деталей товару є однією із ключових. На ній користувач може побачити зображення , опис , характеристики та ціну товару. Сторінка деталей про товар на веб-сайті інтернет-магазину будівельних матеріалів надає користувачам вичерпну інформацію про кожен товар, допомагаючи приймати обґрунтовані рішення щодо купівлі. Розглянемо структуру та функціональність цієї сторінки.

На лівій половині сторінки розташовано слайдер зображень товару. Він

складається з центрального зображення та стрілок для перемикання зображень. Зображення товару відображається за допомогою тега <img>, де URL зображення отримується з моделі товару. Є дві основні кнопки для взаємодії з товаром:

– Купити: Кнопка з червоним фоном, яка відправляє форму з методом "get" на URL для підтвердження купівлі;

– Додати в кошик: Кнопка з прозорим фоном та червоною рамкою, яка відправляє форму з методом "post" на URL для додавання товару до кошика. Тут використовується піктограма кошика з SVG для візуального відображення.

Блок із кнопками дозволяє користувачу вибирати кількість товару для замовлення. Кнопки "+" та "-" змінюють кількість товару, а також є поле для відображення поточної кількості. Код сторінки в додатку А.

3.3.6 Views:

Функція index відображає головну сторінку з усіма товарами. Вона отримує всі товари з бази даних і передає їх у контексті шаблону market/index.html.

```
def index(request):
    products = Product.objects.all()
    context = {
        'products': products,
    }
    return render(request, "market/index.html", context)
```

Функція delete_product видаляє товар за ідентифікатором. Якщо метод запиту - POST, товар видаляється, і користувача перенаправляють на головну сторінку. В іншому випадку відображається сторінка підтвердження видалення market/deleteproduct.html.

```
def delete_product(request, product_id):
```

```
        product = Product.objects.get(id=product_id)
        if request.method == "POST":
            product.delete()
            redirect("/market/")
        context = {'product': product}
                return   render(request,   "market/deleteproduct.html",
context=context)
```

Функція product_detail відображає деталі товару. Вона отримує товар за ідентифікатором, форму для додавання товару до кошика та атрибути товару, після чого передає ці дані у контексті шаблону market/detail.html.

```
    def product_detail(request, product_id):
        product = Product.objects.get(id=product_id)
        form = CartAddProductForm()
        product_attributes =
ProductAttributeValue.objects.filter(product=product)
        return render(request, 'market/detail.html',
                      context={'product': product, 'form': form,
'product_attributes': product_attributes})
```

Функція catalog відображає каталог товарів з можливістю фільтрації за категоріями, брендами, сортуванням та пошуком. Вона також підтримує пагінацію результатів, використовуючи Django Paginator. Результати відображаються у шаблоні market/catalog.html. Код функції в додатку А

Функція aboutUs просто відображає сторінку "Про нас", використовуючи шаблон market/about_us.html.

```
    def aboutUs(request):
        return render(request, 'market/about_us.html')
```

Функція product_attribute_list повертає список атрибутів товару у форматі JSON за категорією. Якщо категорія не задана, повертається порожній список.

```
    def product_attribute_list(request):
        category_id = request.GET.get('category__id')
        if category_id:
                                                    attributes      =
ProductAttribute.objects.filter(category_id=category_id)
             results = [{'id': attr.id, 'text': attr.name} for attr in
attributes]
            return JsonResponse({'results': results})
        return JsonResponse({'results': []})
```

3.3.7 Models

Модель Category використовується для категорій товарів з підтримкою вкладеності. Категорії можуть мати батьківські категорії, що дозволяє створювати ієрархічну структуру. Поле slug автоматично генерується при збереженні, якщо воно не задане вручну.

```
    class Category(models.Model):
        name = models.CharField(max_length=200, db_index=True)
        parent = models.ForeignKey('self', on_delete=models.CASCADE,
related_name='children', null=True, blank=True)
        slug = models.SlugField('URL', max_length=200, unique=True,
db_index=True, null=True, blank=True)
        created_at = models.DateTimeField('Дата створення',
auto_now_add=True)

        class Meta:
            unique_together = (['name', 'parent'])
            verbose_name = 'Категорія'
            verbose_name_plural = 'Категорії'

        def __str__(self):
            full_path = [self.name]
            k = self.parent
            while k is not None:
```

```
                full_path.append(k.name)
                k = k.parent
            return '>'.join(full_path[::-1])

        def save(self, *args, **kwargs):
            if not self.slug:
                self.slug = slugify(rand_slug() + '-pickBetter' +
self.name)
            super(Category, self).save(*args, **kwargs)
```

Модель ProductAttribute визначає атрибути товарів, які належать до певної категорії. Вона зберігає назву атрибута та зв'язок з категорією.

```
    class ProductAttribute(models.Model):
        name = models.CharField(max_length=200)
                    category    =    models.ForeignKey(Category,
related_name='attributes', on_delete=models.CASCADE)

        class Meta:
            verbose_name = 'Характеристика товару'
            verbose_name_plural = 'Характеристики товарів'

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

Модель Product зберігає інформацію про товари, включаючи назву, бренд, зображення, опис, ціну та доступність. Товари можуть належати до категорій, і мають зв'язок з моделью Category.

```
    class Product(models.Model):
                    category    =    models.ForeignKey(Category,
related_name='products', on_delete=models.CASCADE, null=True)
                title   =   models.CharField("назва",   max_length=200,
db_index=True)
         slug = models.SlugField("URL", max_length=200, db_index=True,
null=True, blank=True)
```

```
                brand = models.CharField("бренд", max_length=200,
db_index=True, null=True, blank=True)
                        image = models.ImageField("зображення",
upload_to='products/%Y/%m/%d', blank=True)
        description = models.TextField("опис", null=True, blank=True)
               price = models.DecimalField('ціна', max_digits=10,
decimal_places=2)
        available = models.BooleanField("Наявність", default=True)

        def get_price(self):
            return self.price

        class Meta:
            verbose_name = 'Товар'
            verbose_name_plural = 'Товари'

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

Модель ProductAttributeValue зберігає значення конкретних атрибутів для товарів. Вона має зв'язок з моделями Product та ProductAttribute, що дозволяє зберігати значення атрибутів для кожного товару.

```
    class ProductAttributeValue(models.Model):
                        product = models.ForeignKey(Product,
related_name='attribute_values', on_delete=models.CASCADE)
                attribute = models.ForeignKey(ProductAttribute,
related_name='values', on_delete=models.CASCADE)
        value = models.CharField(max_length=200)

        class Meta:
            verbose_name = 'Значення характеристики товару'
            verbose_name_plural = 'Значення характеристик товарів'

        def __str__(self):
            return f'{self.attribute.name}: {self.value}'
```

3.3.8 Urls

У файлі urls.py визначаються маршрути для основних сторінок додатку. Головна сторінка відображається за допомогою функції index, сторінка "Про нас" - за допомогою функції aboutUs, деталі товару - за допомогою функції product_detail, каталог товарів - за допомогою функції catalog. Також визначається маршрут для каталогу товарів за категорією, який використовує ту ж функцію catalog.

```
urlpatterns = [
    path('', views.index, name='index'),
    path('about-us', views.aboutUs, name='about_us'),
          path('product/<str:product_id>',  views.product_detail,
name="detail"),
    path('catalog/', views.catalog, name='catalog'),
            path('catalog/<str:category_slug>/',   views.catalog,
name='catalog_by_category'),
]
```

3.4 Додаток Users

3.4.1 Views

Функція register обробляє реєстрацію нових користувачів. Якщо метод запиту POST, то форма реєстрації перевіряється на валідність, користувач створюється та автоматично авторизується. Після цього користувача перенаправляють на головну сторінку. Якщо метод запиту GET, відображається порожня форма реєстрації.

```
def register(request):
    if request.method == 'POST':
        form = RegistrationForm(request.POST)
        if form.is_valid():
```

```
                user = form.save()
                username = form.cleaned_data['username']
                password = form.cleaned_data['password1']
                            user  =  authenticate(username=username,
password=password)
                if user is not None:
                    login(request, user)
                    return redirect('/')
        else:
            form = RegistrationForm()
        return render(request, 'users/register.html', {'form': form})
```

Функція create_user_profile створює профіль користувача та кошик після створення нового користувача. Вона викликається автоматично після збереження об'єкта користувача за допомогою сигналу post_save.

```
@receiver(post_save, sender=User)
def create_user_profile(sender, instance, created, **kwargs):
    if created:
        profile = Profile.objects.create(user=instance)
        cart = Cart.objects.create(user=instance)
        profile.cart = cart
        profile.save()
```

Функція user_login обробляє вхід користувача. Якщо метод запиту POST, форма входу перевіряється на валідність, користувач автентифікується та авторизується. Після успішного входу користувача перенаправляють на головну сторінку. Якщо метод запиту GET, відображається порожня форма входу.

```
def user_login(request):
    if request.method == 'POST':
        form = LoginForm(request.POST)
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password1']
                  user  =  authenticate(request,  username=username,
```

```
password=password)
                if user is not None:
                    login(request, user)
                    return redirect('/')
        else:
            form = LoginForm()
        return render(request, 'users/login.html', {'form': form})
```

Функція profile відображає профіль користувача, який містить особисті дані та активні і неактивні замовлення. Активні замовлення сортуються за датою створення у зворотному порядку.

```
    @login_required
    def profile(request):
        user = request.user
        profile = user.profile
        active_orders = Order.objects.filter(user=user,

status__in=['Опрацювання', 'Комплектація', 'Доставляється']).order_by(
            '-created_at')
                inactive_orders = Order.objects.filter(user=user,
status__in=['Отримано покупцем', 'Відмінено']).order_by(
            '-created_at')

        context = {
            'user': user,
            'profile': profile,
            'active_orders': active_orders,
            'inactive_orders': inactive_orders,
        }

        return render(request, 'users/profile.html', context=context)
```

Функція login_required відображає сторінку з повідомленням про необхідність входу у систему для доступу до певних функцій сайту.

```
    def login_required(request):
```

```
        return render(request, 'users/login_required.html')
```

Функція update_profile дозволяє користувачам оновлювати свої профілі. Якщо метод запиту POST, форма оновлення профілю перевіряється на валідність і зберігається. Після цього користувача перенаправляють на сторінку профілю. Якщо метод запиту GET, відображається форма оновлення з поточними даними профілю.

```
    def update_profile(request):
        if request.method == 'POST':
                form = ProfileUpdateForm(request.POST, request.FILES,
instance=request.user.profile)
            if form.is_valid():
                form.save()
                return redirect('users:profile')
        else:
            form = ProfileUpdateForm(instance=request.user.profile)

         return render(request, 'users/profile_update.html', {'form':
form})
```

Функція order_update_profile дозволяє користувачам оновлювати свої профілі під час процесу оформлення замовлення. Працює аналогічно функції update_profile, але після збереження даних перенаправляє користувача на сторінку підтвердження замовлення.

```
    def order_update_profile(request):
        if request.method == 'POST':
                form = ProfileUpdateForm(request.POST, request.FILES,
instance=request.user.profile)
            if form.is_valid():
                form.save()
                return redirect('order:order-confirmation')
        else:
            form = ProfileUpdateForm(instance=request.user.profile)
```

```
            return render(request, 'users/oder_profile_update.html',
{'form': form})
```

Клас CustomPasswordChangeView відповідає за зміну пароля користувача. Він використовує спеціальну форму зміни пароля і перенаправляє користувача на успішну сторінку після зміни пароля.

```
class CustomPasswordChangeView(PasswordChangeView):
    form_class = CustomPasswordChangeForm
    template_name = 'users/change_password.html'
    success_url = reverse_lazy('')
```

Клас CustomPasswordResetView відповідає за скидання пароля користувача. Він відображає форму для введення електронної пошти користувача, надсилає листа зі спеціальним посиланням для скидання пароля і перенаправляє користувача на сторінку успішного скидання.

```
class CustomPasswordResetView(PasswordResetView):
    template_name = 'users/password_reset_form.html'
    email_template_name = 'users/password_reset_email.html'
    success_url = reverse_lazy('users:password_reset_done')
```

Клас CustomPasswordResetDoneView відображає повідомлення про успішне надсилання листа для скидання пароля.

```
class CustomPasswordResetDoneView(PasswordResetDoneView):
    template_name = 'users/password_reset_done.html'
```

Клас CustomPasswordResetConfirmView відповідає за підтвердження скидання пароля. Користувач вводить новий пароль, і після успішного скидання його перенаправляють на відповідну сторінку.

```
class CustomPasswordResetConfirmView(PasswordResetConfirmView):
    template_name = 'users/password_reset_confirm.html'
    success_url = reverse_lazy('users:password_reset_complete')
```

Клас CustomPasswordResetCompleteView відображає повідомлення про успішне скидання пароля після введення нового пароля.

```
class CustomPasswordResetCompleteView(PasswordResetCompleteView):
    template_name = 'users/password_reset_complete.html'
```

3.4.2 Models

Модель Profile зберігає додаткову інформацію про користувача, таку як електронна пошта, контактний номер, ім'я, прізвище, адреса доставки та кошик. Профіль користувача пов'язаний з об'єктом користувача через OneToOneField.

```
class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE,
null=True, blank=True)
    email = models.EmailField(max_length=100, unique=True,
null=True, blank=True)
    contactNumber = models.CharField(max_length=15, default='',
blank=True)
    firstName = models.CharField(max_length=100, blank=True,
null=True)
    lastName = models.CharField(max_length=100, blank=True,
null=True)
    DeliveryAddress = models.CharField(max_length=100, blank=True,
null=True)
    cart = models.OneToOneField(Cart, on_delete=models.SET_NULL,
null=True, blank=True)

    def __str__(self):
        return self.user.username
```

Функція save_user_profile автоматично зберігає об'єкт користувача після збереження профілю за допомогою сигналу post_save.

```
@receiver(post_save, sender=Profile)
def save_user_profile(sender, instance, **kwargs):
    instance.user.save()
```

3.4.3 Urls

Файл urls.py визначає маршрути для основних сторінок додатку. Реєстрація користувачів обробляється функцією register, вхід - через стандартний клас LoginView, вихід - через стандартний клас LogoutView. Також є маршрут для сторінки, яка вимагає входу, оброблюваний функцією login_required.

Профіль користувача відображається за допомогою функції profile, оновлення профілю - за допомогою функцій update_profile та order_update_profile. Зміна пароля обробляється класом CustomPasswordChangeView. Маршрути для скидання пароля включають обробку форм скидання, підтвердження та завершення скидання пароля за допомогою відповідних класів. Код знаходиться в додатку В.

3.5 Додаток Cart

3.5.1 Views

Функція add_to_cart додає товар до кошика користувача. Якщо метод запиту POST, форма додавання товару перевіряється на валідність. Якщо товар вже є в кошику, кількість збільшується, якщо ні - створюється новий запис у моделі CartItem. Після цього користувача перенаправляють на сторінку перегляду кошика.

```
@login_required
def add_to_cart(request, product_id):
    product = get_object_or_404(Product, pk=product_id)
    cart, created = Cart.objects.get_or_create(user=request.user)
    if request.method == 'POST':
        form = CartAddProductForm(request.POST)
        if form.is_valid():
            quantity = form.cleaned_data['quantity']
                                        cart_item,   created   =
CartItem.objects.get_or_create(cart=cart, product=product)
            if not created:
                cart_item.quantity += quantity
            else:
                cart_item.quantity = quantity
            cart_item.save()
            return redirect('cart:cart-view')
    else:
        form = CartAddProductForm()
    return render(request, 'cart/cart_view.html', {'form': form})
```

Функція cart відображає поточний стан кошика користувача. Вона отримує всі товари в кошику, обчислює загальну вартість та передає ці дані у контекст для відображення на сторінці.

```
@login_required
def cart(request):
    cart, created = Cart.objects.get_or_create(user=request.user)
    cart_items = cart.cartitem_set.all()
    total_price = cart.get_total_price()
    context = {
        'cart': cart,
        'cart_items': cart_items,
        'total_price': total_price,
    }
    return render(request, 'cart/cart_view.html', context)
```

Функція remove_from_cart видаляє товар з кошика. Після видалення

користувача перенаправляють на сторінку перегляду кошика.

```
@login_required
def remove_from_cart(request, product_id):
    cart = Cart.objects.get(user=request.user)
    product = Product.objects.get(pk=product_id)
    cart_item = CartItem.objects.get(cart=cart, product=product)
    cart_item.delete()
    return redirect('cart:cart-view')
```

Функція update_cart оновлює кількість товару в кошику. Якщо метод запиту POST, залежно від дії (збільшення або зменшення кількості), кількість товару оновлюється. Якщо кількість товару зменшується до одного і далі, товар видаляється з кошика. Після оновлення користувача перенаправляють на сторінку перегляду кошика.

```
@login_required
def update_cart(request, cart_item_id):
    cart_item = get_object_or_404(CartItem, pk=cart_item_id)
    if request.method == 'POST':
        action = request.POST.get('action')
        if action == 'increase':
            cart_item.quantity += 1
        elif action == 'decrease' and cart_item.quantity > 1:
            cart_item.quantity -= 1
        elif action == 'decrease' and cart_item.quantity == 1:
            cart_item.delete()
            return redirect('cart:cart-view')
        cart_item.save()
    return redirect('cart:cart-view')
```

3.5.2 Models

Модель Cart представляє кошик користувача. Вона пов'язана з користувачем через OneToOneField і може містити багато продуктів через

модель CartItem. Метод get_total_price обчислює загальну вартість товарів у кошику, перебираючи всі елементи кошика та додаючи ціну кожного товару з урахуванням кількості.

```
class Cart(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, related_name='cart')
    products = models.ManyToManyField(Product, through='CartItem')

    def get_total_price(self):
        total_price = 0
        for item in self.cartitem_set.all():
            total_price += item.product.get_price() * item.quantity
        return total_price
```

Модель CartItem представляє окремий товар у кошику. Вона пов'язана з моделями Product і Cart через ForeignKey. Поле quantity зберігає кількість одиниць товару в кошику. Метод get_product_price обчислює загальну вартість товару в кошику, множачи ціну товару на кількість.

```
class CartItem(models.Model):
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    cart = models.ForeignKey(Cart, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=1)

    def get_product_price(self):
        return self.product.price * self.quantity
```

3.5.3 Urls

Файл urls.py визначає маршрути для основних дій з кошиком. Додавання товару до кошика обробляється функцією add_to_cart, видалення - функцією remove_from_cart, перегляд кошика - функцією cart, оновлення кількості товару

- функцією update_cart.

```
app_name = 'cart'

urlpatterns = [
        path('add-to-cart/<int:product_id>/',  views.add_to_cart, name='add_to_cart'),
        path('remove/<int:product_id>/',  views.remove_from_cart, name='remove_from_cart'),
    path('', views.cart, name='cart-view'),
          path('update/<int:cart_item_id>/',  views.update_cart, name='update_cart'),
]
```

3.6 Додаток Order

Order - це додаток, відповідальний за обробку замовлень у веб-додатку BudMarket. Він забезпечує функціонал створення замовлень, відстеження їх статусу та зберігання інформації про замовлення.

3.6.1 Views

Функція process_order обробляє створення нового замовлення на основі товарів, що знаходяться у кошику користувача. Якщо метод запиту POST, створюється нове замовлення, товари додаються до нього, і користувача перенаправляють на сторінку з деталями замовлення або сторінку оплати. Якщо метод запиту GET, відображається сторінка з кошиком користувача. Код функції знаходиться в додатку Б

Функція order_detail відображає детальну інформацію про конкретне замовлення. Вона приймає ідентифікатор замовлення в якості параметра і повертає HTML сторінку з інформацією про замовлення, включаючи товари, ціну та статус.

```
def order_detail(request, order_id):
```

```
        order = get_object_or_404(Order, pk=order_id)
        if order.user != request.user:
            return HttpResponseForbidden("You don't have permission to
view this order.")
          return render(request, 'order/order_detail.html', {'order':
order})
```

Функція cancel_order дозволяє користувачу скасувати замовлення. Якщо метод запиту POST, замовлення скасовується, і користувача перенаправляють на сторінку з підтвердженням скасування або на сторінку деталей замовлення. Якщо метод запиту GET, відображається форма підтвердження скасування.

```
    def cancel_order(request, order_id):
        order = get_object_or_404(Order, pk=order_id)
        if request.method == 'POST':
            order.status = Order.CANCELED
            order.save()
            messages.info(request, "Замовлення скасовано.")
            return redirect('order:order-detail', order_id=order_id)
         return render(request, 'order/cancel_order.html', {'order':
order})
```

Функція save_transaction_id зберігає ідентифікатор транзакції для замовлення. Вона викликається платіжною системою після обробки платежу і повертає JSON відповідь зі статусом операції.

```
    @csrf_exempt
    def save_transaction_id(request):
        if request.method == 'POST':
            data = json.loads(request.body)
            transaction_id = data.get('transactionId')
            order = Order.objects.filter(id=transaction_id).first()
            if order:
                order.payment_id = transaction_id
                order.save()
                           return JsonResponse({'status': 'success',
```

```
'transaction_id': transaction_id})
              return JsonResponse({'status': 'error', 'message': 'Order
not found'}, status=404)
         return JsonResponse({'status': 'error'}, status=400)
```

Функція order_confirm відображає сторінку підтвердження замовлення з підсумком кошика. Вона показує інформацію про товари в кошику, загальну вартість замовлення та форму для введення інформації про доставку та оплату.

```
     def order_confirm(request):
         if not request.user.is_authenticated:
               return redirect('login')  # Перенаправлення на сторінку
логіну, якщо користувач не авторизований
         cart = Cart.objects.filter(user=request.user).first()
         if not cart:
             messages.error(request, "Кошик порожній.")
             return redirect('cart:cart-view')
         cart_items = cart.cartitem_set.all()
          total_price = sum(item.product.price * item.quantity for item
in cart_items)

         context = {
             'cart_items': cart_items,
             'total_price': total_price,
         }
         return render(request, 'order/confirmation.html', context)
```

Функція confirm_buy_now обробляє покупку товару безпосередньо, минаючи кошик. Якщо метод запиту POST, створюється нове замовлення, товар додається до нього, і користувача перенаправляють на сторінку успіху або повертається JSON відповідь з секретом клієнта для Stripe. Якщо метод запиту GET, відображається форма підтвердження покупки. Код функції знаходиться в додатку Г.

Функція order_success відображає сторінку успішного оформлення замовлення. Вона показує повідомлення про успішне замовлення та основну

інформацію про нього.

```
@login_required
def order_success(request):
    return render(request, 'order/success.html')
```

Функція stripe_webhook обробляє веб-хуки від Stripe для підтвердження платежів. Вона приймає HTTP запит від Stripe, перевіряє його на валідність та обробляє зміну статусу платежу.

```
@csrf_exempt
def stripe_webhook(request):
    payload = request.body
    sig_header = request.META['HTTP_STRIPE_SIGNATURE']
    endpoint_secret = settings.STRIPE_ENDPOINT_SECRET
    try:
        event = stripe.Webhook.construct_event(
            payload, sig_header, endpoint_secret
        )
    except ValueError as e:
        return HttpResponse(status=400)
    except stripe.error.SignatureVerificationError as e:
        return HttpResponse(status=400)
    if event['type'] == 'payment_intent.succeeded':
        payment_intent = event['data']['object']
        order_id = payment_intent['metadata']['order_id']
        order = Order.objects.get(id=order_id)
        order.paid = True
        order.save()
    return HttpResponse(status=200)
```

3.6.2 Models

Клас Order представляє модель замовлення, яка містить основну інформацію про замовлення, включаючи користувача, товари, дату створення,

загальну вартість, статус, метод доставки та оплати, адресу доставки, ідентифікатор платіжного наміру Stripe та статус оплати. Код класу знаходиться в додатку Г.

Клас OrderItem представляє модель товару в замовленні. Він містить інформацію про товар, кількість та ціну.

```
class OrderItem(models.Model):
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    order = models.ForeignKey(Order, on_delete=models.CASCADE)
    quantity = models.PositiveIntegerField(default=1)
    price = models.DecimalField(max_digits=10, decimal_places=2)
```

Клас Payment представляє модель платежу, яка зберігає інформацію про суму платежу, дату та час, статус успішності платежу та ідентифікатор платіжного наміру Stripe.

```
class Payment(models.Model):
    amount = models.DecimalField(max_digits=8, decimal_places=2)
    timestamp = models.DateTimeField(auto_now_add=True)
    success = models.BooleanField(default=False)
    stripe_payment_intent = models.CharField(max_length=255)
    def __str__(self):
              return f"Payment {'success' if self.success else 'failure'} on {self.timestamp.strftime('%Y-%m-%d %H:%M')}"
```

3.6.3 Urls

Маршрути додатку order включають: order-confirm для відображення сторінки підтвердження замовлення; process-order для обробки процесу оформлення замовлення; order-detail для відображення деталей замовлення; cancel-order для скасування замовлення; save-transaction для збереження ідентифікатора транзакції; confirm-buy-now для обробки безпосередньої покупки товару; stripe-webhook для обробки веб-хуків від Stripe, та order-success

для відображення сторінки успішного оформлення замовлення. Код знаходиться в додатку Г.

Висновки до розділу 3

У третьому розділі кваліфікаційної роботи було здійснено практичну реалізацію веб-сайту інтернет-магазину будівельних матеріалів. Основні результати включають реалізацію основних функціональних можливостей, зокрема, можливість перегляду каталогу товарів, сортування товарів за різними параметрами, додавання товарів до кошика, оформлення замовлень, а також створення та управління профілями користувачів. Для реалізації сайту використовувались мови програмування та фреймворки, такі як Python, HTML, CSS, JavaScript, а також Django і TailwindCSS. Це забезпечило високу швидкість завантаження сторінок та зручність користування сайтом. Було створено окремі додатки для різних частин сайту (Market, Users, Cart, Order), що дозволило чітко структурувати код та спростити його подальше обслуговування та розширення. Особлива увага приділялась створенню зручного та естетично привабливого інтерфейсу користувача. Було враховано сучасні тенденції веб-дизайну та адаптивності, що забезпечує комфортне використання сайту на різних пристроях. Додатково реалізовано такі функції, як система пошуку та фільтрації товарів, механізм відновлення паролю, а також інформування користувачів про статус їх замовлень. Таким чином, у третьому розділі було реалізовано повнофункціональний веб-сайт інтернет-магазину, який відповідає сучасним вимогам та забезпечує зручність використання як для покупців, так і для адміністраторів сайту

ВИСНОВОК

У ході виконання кваліфікаційної роботи було розроблено та впроваджено веб-додаток BudMarket для ефективного управління онлайн-магазином будівельних матеріалів. Застосування сучасних технологій та фреймворків, таких як Django, дозволило створити надійну та масштабовану платформу. Основні функціональні можливості включають авторизацію та реєстрацію користувачів, перегляд каталогу товарів, додавання товарів до кошика, оформлення замовлень, а також інтеграцію платіжної системи для здійснення онлайн-платежів.

Серед переваг розробленої системи можна виділити зручний інтерфейс, високу швидкодію, безпеку даних користувачів та можливість адміністрування через спеціальну панель. Крім того, система забезпечує зручну навігацію для користувачів, що сприяє підвищенню їх задоволеності та лояльності.

В процесі роботи було розглянуто та проаналізовано аналогічні рішення, що дозволило виокремити найкращі практики та уникнути поширених недоліків. Реалізація проекту продемонструвала важливість комплексного підходу до розробки програмного забезпечення, включаючи етапи планування, проектування, кодування, тестування та впровадження.

Таким чином, створена веб-платформа BudMarket відповідає сучасним вимогам ринку та готовий до подальшого розширення та вдосконалення, що відкриває нові перспективи для розвитку бізнесу замовника.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rozetka: веб-сайт. URL: https://rozetka.com.ua (дата звернення: 24.02.2024)

2. Епіцентр: веб-сайт. URL: https://epicentrk.ua (дата звернення 24.02.2024)

3. Нова Лінія: веб-сайт. URL: www.nl.ua (дата звернення 25.02.2024)

4. Python: веб-сайт. URL: https://www.python.org (дата звернення 25.02.2024)

5. Django: веб-сайт. URL: https://www.djangoproject.com (дата звернення 25.02.2024)

6. SQLite: веб-сайт. URL: https://www.djangoproject.com (дата звернення 25.02.2024)

7. Diagrams.net: веб-сайт URL: https://app.diagrams.net (дата звернення 04.03.2024)

8. Figma: веб-сайт URL: https://www.figma.com (дата звернення 04.03.2024)

9. Stripe: веб-сайт URL: https://stripe.com (дата звернення 26.03.2024)

10. PyCharm: веб-сайт URL: https://www.jetbrains.com/pycharm/ (дата звернення 04.03.2024)

11. HTML: веб-сайт URL: https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML (дата звернення 04.03.2024)

12. CSS: веб-сайт URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення 04.03.2024)

13. JavaScript: веб-сайт. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення 12.05.2024)

14. TailwindCSS : веб-сайт. URL : https://www.postgresql.org/ (дата

звернення 15.05.2024)

15. Офіційна документація TailwindCSS : веб-сайт. URL : https://tailwindcss.com/docs/ (дата звернення 04.03.2024)

16. PostgreSQL: веб-сайт. https://docs.djangoproject.com/en/5.0/ (дата звернення 30.05.2024)

17. Офіційна документація Django : веб-сайт. https://docs.djangoproject.com/en/5.0/ (дата звернення 30.05.2024)

18. GitHub репозиторій : веб-сайт. URL: https://github.com/NickNameSOON/BudMarket (дата звернення 30.05.2024)

19. Wireframe проекту: веб-сайт. URL: https://www.figma.com/design/Ll13jLIX7GREOBcbTkqDWf/Budmarket?node-id=0-1&t=Dm2rdIdTWekbznJC-1 (дата звернення 30.05.2024)

20. Ngrok: веб-сайт. URL: https://ngrok.com/ (дата звернення 30.05.2024)

ДОДАТКИ

Додаток А

Код додатку Market

```
<!DOCTYPE html>
{% extends 'market/base.html' %}
{% block content %}
    <head>
        <title>БудМаркет</title>
    </head>
    <body class="bg-gray-100">
    <main class="container mx-auto px-4 py-8">
        <h1 class="text-3xl text-center font-bold mb-2">Покращіть
своє житло <span class="text-red-600">за вигідними цінами</span>
        </h1>
        <p class="text-lg text-center mb-6">Купуйте найкраще для
вашої оселі</p>
        <h2 class="text-2xl space-x-4 font-semibold ml-44
mb-4">Рекомендуємо до покупки:</h2>
        <div class="flex justify-center items-center space-x-3">
            <button id="scroll-left" class="focus:outline-none">
                <svg class="w-6 h-6 text-gray-500"
fill="currentColor" xmlns="http://www.w3.org/2000/svg"
                     viewBox="0 0 24 24">
                    <path d="M15.41 7.41L14 6l-6 6 6 6
1.41-1.41L10.83 12z"/>
                </svg>
            </button>
            <div id="products-carousel" class="flex space-x-4
overflow-x-scroll scrollbar-hide w-full max-w-6xl">
                {% for product in products %}
                    <a href="{% url 'market:detail' product.id %}"
class="min-w-[16.66%] flex-shrink-0">
                        <div class="bg-white p-4 rounded-lg
shadow-md w-[308px] h-[323px] flex flex-col justify-between">
```

```
                                <img src="{{ product.image.url }}"
class="h-32 bg-gray-300 rounded-md mb-2">
                                <h3 class="text-xl font-semibold
truncate-text">{{ product.title }}</h3>
                                <p class="truncate-text">Виробник: {{
product.brand }}</p>
                                <p class="text-green-500 text-lg
font-bold">{{ product.price }}</p>
                            </div>
                        </a>
                    {% empty %}
                        <div class="text-4xl font-extrabold
text-gray-900">No products found</div>
                    {% endfor %}
                </div>
                <button id="scroll-right" class="focus:outline-none">
                    <svg class="w-6 h-6 text-gray-500"
fill="currentColor" xmlns="http://www.w3.org/2000/svg"
                         viewBox="0 0 24 24">
                        <path d="M8.59 16.59L10 18l6-6-6-6-1.41
1.41L13.17 12z"/>
                    </svg>
                </button>
            </div>
        </main>

        <script>
            const scrollContainer =
document.getElementById('products-carousel');
            const scrollAmount = scrollContainer.clientWidth;

            document.getElementById('scroll-left').onclick = function
() {
                scrollContainer.scrollBy({
                    left: -scrollAmount,
                    behavior: 'smooth'
                });
            };
```

```
            document.getElementById('scroll-right').onclick = function
() {
                scrollContainer.scrollBy({
                    left: scrollAmount,
                    behavior: 'smooth'
                });
            };
        </script>
        </body>
    {% endblock content %}

    <!DOCTYPE html>
    {% extends "market/base.html" %}
    {% load static %}

    {% block content %}
        <html lang="uk">
        <head>
            <meta charset="UTF-8">
            <meta name="viewport" content="width=device-width,
initial-scale=1.0">
            <title>{{ product.title }}</title>
            <link
href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min
.css" rel="stylesheet">
            <!-- Include SweetAlert2 library -->
            <script
src="https://cdn.jsdelivr.net/npm/sweetalert2@11"></script>
        </head>
        <body class="bg-gray-100">
        <div class="container mx-auto p-4">
            <div class="flex">
                <!-- Product Image Slider -->
                <div class="w-1/2 p-4 flex items-center">
                    <div class="text-4xl text-gray-500
mr-2">&lt;</div>
                    <img src="{{ product.image.url }}"
class="bg-gray-200 h-96 flex items-center justify-center rounded">
                    <!-- Image slider placeholder -->
```

```
                    <div class="text-4xl text-gray-500 ml-2">
&gt;</div>
                </div>

                <!-- Product Details -->
                <div class="w-1/2 p-4">
                    <h1 class="text-4xl font-bold mb-4">{{
product.title }}</h1>
                    <p class="text-gray-700 text-lg
mb-2"><strong>Виробник:</strong> {{ product.brand }}</p>
                    <p class="text-gray-700 text-lg
mb-2"><strong>Опис:</strong> {{ product.description }}</p>
                    <p class="text-gray-700 text-lg
mb-4"><strong>Характеристики:</strong>
                        {% for attribute_value in
product.attribute_values.all %}
                            <li>{{ attribute_value.attribute.name }}:
{{ attribute_value.value }}</li>
                        {% endfor %}</p>

                    <div class="text-4xl text-green-600 font-bold mt-6
mb-6">{{ product.price }} ₴</div>

                    <div class="flex items-center mb-6">
                        <button class="bg-green-500 text-white p-2
rounded-l">-</button>
                        <span class="border px-4 py-2">1</span>
                        <button class="bg-green-500 text-white p-2
rounded-r">+</button>
                    </div>

                    <div class="flex items-center">
                        <form method="get" action="{% url
'order:confirm-buy-now' product.id %}">
                            <button type="submit" class="bg-red-500
text-white py-2 px-4 rounded mr-4">Купити</button>
                        </form>
                        <form method="post" action="{% url
'cart:add_to_cart' product.id %}" id="add-to-cart-form">
```

```
                                {% csrf_token %}
                                <input type="hidden" name="quantity"
value="1"> <!-- Додайте це поле для кількості -->
                                <button type="submit"
                                        class="bg-transparent border
border-red-500 text-red-500 py-2 px-4 rounded">
                                    <svg class="w-6 h-6 inline-block mr-2"
fill="none" stroke="currentColor" viewBox="0 0 24 24"

xmlns="http://www.w3.org/2000/svg">
                                        <path stroke-linecap="round"
stroke-linejoin="round" stroke-width="2"
                                              d="M3 3h2l.34 2M7
13h10l4-8H5.34M7 13l-1.34 4h14.68M7 13H5m5 0a4 4 0 118 0m-4
4v2m0-2h-2m2 0h2"></path>
                                    </svg>
                                    Додати в кошик
                                </button>
                            </form>

                        </div>
                    </div>
                </div>
            </div>

            <!-- JavaScript code for handling button click event -->
            </body>
            </html>
        {% endblock content %}

        <!DOCTYPE html>
        {% extends "market/base.html" %}

        {% block content %}
            <html lang="uk">
            <head>
                <title>Каталог</title>
            </head>
            <body class="bg-gray-100">
```

```
        <div class="container mx-auto p-4">
            <div class="flex">
                <!-- Sidebar Filters -->
                <form method="get" action="{% url 'market:catalog' %}"
class="w-1/4 p-4 bg-white rounded shadow-md">
                    <h2 class="text-lg font-semibold mb-4">Сортувати
за:</h2>
                    <div class="mb-4">
                        <div class="flex items-center mb-1">
                            <input type="radio" id="no_sort"
name="sort" value="" class="mr-2"
                                   {% if not sort_param %}checked{%
endif %}>
                            <label for="no_sort">Без
сортування</label>
                        </div>
                        <div class="flex items-center mb-1">
                            <input type="radio" id="min_price"
name="sort" value="min_price" class="mr-2"
                                   {% if sort_param == 'min_price'
%}checked{% endif %}>
                            <label for="min_price">Мінімальна
ціна</label>
                        </div>
                        <div class="flex items-center mb-1">
                            <input type="radio" id="max_price"
name="sort" value="max_price" class="mr-2"
                                   {% if sort_param == 'max_price'
%}checked{% endif %}>
                            <label for="max_price">Максимальна
ціна</label>
                        </div>
                    </div>

                    <h2 class="text-lg font-semibold
mb-4">Категорії</h2>
                    <div class="mb-4">
                        {% for c in categories %}
                            <div class="flex items-center">
```

```
                                <input type="radio" id="{{ c.slug }}"
name="category" value="{{ c.slug }}"
                                       class="mr-2" {% if c.slug ==
category.slug %}checked{% endif %}>
                                <label for="{{ c.slug }}">{{ c.name
}}</label>
                            </div>
                        {% endfor %}
                    </div>

                    <h2 class="text-lg font-semibold mb-4">Бренди</h2>
                    <div class="mb-4">
                        {% for brand in brands %}
                            <div class="flex items-center">
                                <input type="checkbox" id="brand-{{
brand.brand }}" name="brand"
                                       value="{{ brand.brand }}"
class="mr-2"
                                        {% if brand.brand in
selected_brands %} checked {% endif %}>
                                <label for="brand-{{ brand.brand
}}">{{ brand.brand }} ({{ brand.count }})</label>
                            </div>
                        {% endfor %}
                    </div>

                    <button class="w-full bg-red-500 text-white py-2
rounded">Застосувати</button>
                    <a href="{% url 'market:catalog' %}" class="block
mt-4 text-red-500 text-center">Скинути</a>
                </form>

                <!-- Main Content -->
                <div class="w-3/4 p-4">
                    <!-- Search Form -->
                    <form method="get" action="{% url 'market:catalog'
%}" class="mb-6">
                        <input type="text" name="q"
placeholder="Пошук" value="{{ query }}"
```

```
                                class="w-full p-2 border
border-gray-300 rounded">
                        <button type="submit"
class="hidden">Пошук</button>
                    </form>

                    <!-- Product Grid -->
                    <h1 class="text-3xl text-center font-bold
mb-6">Каталог</h1>
                    <div class="grid grid-cols-4 gap-4">
                        <!-- Loop through products and create a card
for each one -->
                        {% for product in products %}
                            <div class="bg-white p-4 items-center
rounded shadow-md">
                                <img src="{{ product.image.url }}"
alt="{{ product.title }}" class="bg-gray-200 h-40 mb-4">
                                <h2 class="text-xl font-semibold
mb-2">{{ product.title }}</h2>
                                <p class="text-gray-600
mb-4">Виробник: {{ product.brand }}</p>
                                <p class="text-green-600 text-lg
font-bold">{{ product.price }} ₴</p>
                            </div>
                        {% endfor %}
                    </div>
                    <div class="pagination text-center mt-6">
                    <span class="step-links">
                        {% if products.has_previous %}
                            <a href="?page=1{% if query %}&q={{ query
}}{% endif %}">&laquo; перша</a>
                            <a href="?page={{
products.previous_page_number }}{% if query %}&q={{ query }}{% endif
%}">попередня</a>
                        {% endif %}

                        <span class="current">
                            Сторінка {{ products.number }} з {{
products.paginator.num_pages }}
```

```
                        </span>

                        {% if products.has_next %}
                            <a href="?page=
                                    {{ products.next_page_number }}{%
if query %}&q={{ query }}{% endif %}">наступна</a>
                            <a href="?page={{
products.paginator.num_pages }}{% if query %}&q={{ query }}{% endif
%}">остання &raquo;</a>
                        {% endif %}
                    </span>
                    </div>
                </div>
            </div>
        </div>
        </body>
        </html>
    {% endblock content %}

    {% extends "market/base.html" %}

    {% block content %}
        <head>
            <title>Про нас - БудМаркет</title>
        </head>
        <body class="bg-gray-100">
        <main class="max-w-7xl items-center mx-auto py-8 px-4">
            <section class="text-center mb-12">
                <h1 class="text-4xl font-bold mb-4">Про нас</h1>
                <p class="text-lg mb-4">БудМаркет - це магазин
будівельних та господарських товарів, який працює більше 10
                    років</p>
                <p class="text-md">БудМаркет - це магазин будівельних
та господарських товарів, який працює більше 10 років.
                    Ми пропонуємо широкий асортимент продукції для
ремонту, будівництва та облаштування дому. У нашому
                    магазині ви знайдете все необхідне: від
будівельних матеріалів і інструментів до товарів для саду та
                    городу.
```

```
                    Ми пишаємося високою якістю нашої продукції та
відмінним обслуговуванням клієнтів. Наша команда фахівців
                    завжди готова допомогти вам у виборі потрібних
товарів і надати професійні консультації.

                    Завдяки багаторічному досвіду ми знаємо, як
важливо забезпечити клієнтів всім необхідним для успішної
                    реалізації їхніх проектів. БудМаркет - це надійний
партнер у будь-яких будівельних та господарських
                    питаннях. Завітайте до нас і переконайтеся в цьому
самі!</p>
            </section>
            <h2 class="text-center text-3xl font-bold mb-6">Контактні
дані</h2>
            <section class="text-center mb-12">
                <div class="items-center flex px-12 justify-between">
                    <div class="flex flex-col">
                        <h3 class="text-xl font-semibold mb-2">Графік
роботи</h3>
                        <p>Пн-Пт - 9:00-18:00</p>
                        <p>Сб - 10:00-17:00</p>
                        <p>Нд - Вихідний</p>
                    </div>
                    <div class="items-center mb-6">
                        <iframe
src="https://www.google.com/maps/embed?pb=!1m14!1m12!1m3!1d662.67620793
3098!2d25.0347481134202 57!3d48.366195202652925!2m3!1f0!2f0!3f0!3m2!1i10
24!2i768!4f13.1!5e0!3m2!1suk!2sua!4v1716586889598!5m2!1suk!2sua"
                                width="400" height="200"
style="border:0;" allowfullscreen="" loading="lazy"

referrerpolicy="no-referrer-when-downgrade"></iframe>
                    </div>
                    <div class="flex flex-col">
                        <h3 class="text-xl font-semibold
mb-2">Контакти</h3>
                        <a href="tel:+380980178968" class="flex
text-blue-500 hover:underline">
```

```
                                <svg class="mr-3" width="24" height="24"
viewBox="0 0 24 24" fill="none"
                                     xmlns="http://www.w3.org/2000/svg">
                                    <path d="M12 2C6.47715 2 2 6.47715 2
12C2 17.5229 6.47715 22 12 22C17.5229 22 22 17.5229 22 12C22 6.47715
17.5229 2 12 2ZM8.40625 5.97217C8.61228 5.96002 8.79852 6.08317 8.9397
6.30175L10.3069 8.89452C10.4508 9.20175 10.369 9.53072 10.1543
9.75023L9.52808 10.3765C9.48943 10.4294 9.464 10.4891 9.46338
10.5547C9.70353 11.4843 10.432 12.3417 11.0747 12.9314C11.7174 13.5211
12.4082 14.3194 13.3049 14.5085C13.4158 14.5395 13.5516 14.5505 13.6309
14.4768L14.3584 13.7359C14.6095 13.5455 14.9728 13.4533 15.241
13.6089H15.2532L17.7202 15.0652C18.0823 15.2922 18.1198 15.7309 17.8606
15.9978L16.1614 17.6836C15.9104 17.9409 15.5771 18.0275 15.2532
18.0278C13.8207 17.9849 12.4672 17.2819 11.3555 16.5593C9.53067 15.2318
7.85678 13.5852 6.80592 11.596C6.40288 10.7618 5.92943 9.69745 5.97462
8.76638C5.97865 8.41612 6.07342 8.07297 6.32008 7.8472L8.0193
6.14798C8.15167 6.03535 8.28262 5.97947 8.40625 5.97217Z"
                                          fill="#3C3C3E"></path>
                                </svg>
                                +380 980 178 68</a>
                            <a href="https://t.me/savchuk_28" class="flex
text-blue-500 hover:underline">
                                <svg class="mr-3" width="25" height="24"
viewBox="0 0 25 24" fill="none"
                                     xmlns="http://www.w3.org/2000/svg">
                                    <path d="M12.5 2C6.98 2 2.5 6.48 2.5
12C2.5 17.52 6.98 22 12.5 22C18.02 22 22.5 17.52 22.5 12C22.5 6.48
18.02 2 12.5 2ZM17.14 8.8C16.99 10.38 16.34 14.22 16.01 15.99C15.87
16.74 15.59 16.99 15.33 17.02C14.75 17.07 14.31 16.64 13.75 16.27C12.87
15.69 12.37 15.33 11.52 14.77C10.53 14.12 11.17 13.76 11.74 13.18C11.89
13.03 14.45 10.7 14.5 10.49C14.5069 10.4582 14.506 10.4252 14.4973
10.3938C14.4886 10.3624 14.4724 10.3337 14.45 10.31C14.39 10.26 14.31
10.28 14.24 10.29C14.15 10.31 12.75 11.24 10.02 13.08C9.62 13.35 9.26
13.49 8.94 13.48C8.58 13.47 7.9 13.28 7.39 13.11C6.76 12.91 6.27 12.8
6.31 12.45C6.33 12.27 6.58 12.09 7.05 11.9C9.97 10.63 11.91 9.79 12.88
9.39C15.66 8.23 16.23 8.03 16.61 8.03C16.69 8.03 16.88 8.05 17
8.15C17.1 8.23 17.13 8.34 17.14 8.42C17.13 8.48 17.15 8.66 17.14 8.8Z"
                                          fill="#3C3C3E"></path>
                                </svg>
```

```
                                @savchuk_28</a>
                            <a href="https://www.facebook.com/savchuk_28"
class="flex text-blue-500 hover:underline">
                                <svg class="mr-3" width="20" height="20"
viewBox="0 0 20 20" fill="none"
                                     xmlns="http://www.w3.org/2000/svg">
                                    <path d="M20 10C20 4.48 15.52 0 10
0C4.48 0 0 4.48 0 10C0 14.84 3.44 18.87 8 19.8V13H6V10H8V7.5C8 5.57
9.57 4 11.5 4H14V7H12C11.45 7 11 7.45 11 8V10H14V13H11V19.95C16.05
19.45 20 15.19 20 10Z"
                                          fill="#3C3C3E"></path>
                                </svg>
                                savchuk_28</a>
                            <a href="viber://chat?number=+380980178968"
class="flex text-blue-500 hover:underline">
                                <svg class="mr-3" width="24" height="24"
viewBox="0 0 24 24" fill="none"
                                     xmlns="http://www.w3.org/2000/svg">
                                    <path fill-rule="evenodd"
clip-rule="evenodd"
                                          d="M16.6759 2.62805C13.5297
1.92477 10.2671 1.92477 7.12088 2.62805L6.78188 2.70305C5.8969 2.90091
5.08409 3.34023 4.43375 3.97221C3.78341 4.60419 3.321 5.40409 3.09788
6.28305C2.30038 9.42557 2.30038 12.7175 3.09788 15.86C3.31066 16.6983
3.74129 17.4654 4.34612 18.0836C4.95094 18.7018 5.70846 19.149 6.54188
19.38L7.00688 22.156C7.02173 22.2442 7.05993 22.3268 7.11752
22.3951C7.1751 22.4635 7.24997 22.5152 7.33431 22.5448C7.41865 22.5744
7.50939 22.5809 7.59708 22.5635C7.68476 22.5461 7.76619 22.5056 7.83288
22.446L10.5639 20.003C12.614 20.1282 14.6715 19.9642 16.6759
19.516L17.0159 19.441C17.9009 19.2432 18.7137 18.8039 19.364
18.1719C20.0144 17.5399 20.4768 16.74 20.6999 15.861C21.4973 12.7185
21.4973 9.42658 20.6999 6.28405C20.4767 5.40496 20.0141 4.60498 19.3636
3.97298C18.7131 3.34099 17.9 2.90174 17.0149 2.70405L16.6759
2.62805ZM7.96488 6.20205C7.77903 6.17499 7.58952 6.2124 7.42788
6.30805H7.41388C7.03888 6.52805 6.70088 6.80505 6.41288 7.13105C6.17288
7.40805 6.04288 7.68805 6.00888 7.95805C5.98888 8.11805 6.00288 8.28005
6.04988 8.43304L6.06788 8.44305C6.33788 9.23604 6.68988 9.99904 7.11988
10.717C7.67433 11.7255 8.35664 12.6582 9.14988 13.492L9.17388
13.526L9.21188 13.554L9.23488 13.581L9.26288 13.605C10.0996 14.4007
```

11.0345 15.0862 12.0449 15.645C13.1999 16.274 13.9009 16.571 14.3219
16.695V16.701C14.4449 16.739 14.5569 16.756 14.6699 16.756C15.0284
16.7295 15.3678 16.5838 15.6339 16.342C15.9589 16.054 16.2339 15.715
16.4479 15.338V15.331C16.6489 14.951 16.5809 14.593 16.2909
14.35C15.7084 13.8411 15.0785 13.389 14.4099 13C13.9619 12.757 13.5069
12.904 13.3229 13.15L12.9299 13.646C12.7279 13.892 12.3619 13.858
12.3619 13.858L12.3519 13.864C9.62088 13.167 8.89188 10.402 8.89188
10.402C8.89188 10.402 8.85788 10.026 9.11088 9.83404L9.60288
9.43805C9.83888 9.24605 10.0029 8.79205 9.74988 8.34405C9.36349 7.67481
8.91232 7.04512 8.40288 6.46405C8.29177 6.32729 8.13595 6.23415 7.96288
6.20105L7.96488 6.20205ZM12.5789 5.00005C12.4463 5.00005 12.3191
5.05272 12.2253 5.14649C12.1316 5.24026 12.0789 5.36744 12.0789
5.50005C12.0789 5.63265 12.1316 5.75983 12.2253 5.8536C12.3191 5.94737
12.4463 6.00005 12.5789 6.00005C13.8439 6.00005 14.8939 6.41305 15.7249
7.20505C16.1519 7.63805 16.4849 8.15105 16.7029 8.71305C16.9219 9.27605
17.0219 9.87704 16.9959 10.479C16.9931 10.5447 17.0033 10.6103 17.0259
10.672C17.0485 10.7337 17.083 10.7904 17.1275 10.8387C17.2173 10.9365
17.3423 10.9945 17.4749 11C17.6075 11.0056 17.7369 10.9583 17.8346
10.8684C17.9323 10.7786 17.9903 10.6537 17.9959 10.521C18.0269 9.78053
17.904 9.04163 17.6349 8.35105C17.3646 7.65718 16.9548 7.02616 16.4309
6.49705L16.4209 6.48705C15.3899 5.50205 14.0849 5.00005 12.5789
5.00005ZM12.5449 6.64405C12.4123 6.64405 12.2851 6.69672 12.1913
6.79049C12.0976 6.88426 12.0449 7.01144 12.0449 7.14405C12.0449 7.27665
12.0976 7.40383 12.1913 7.4976C12.2851 7.59137 12.4123 7.64405 12.5449
7.64405H12.5619C13.4739 7.70905 14.1379 8.01305 14.6029 8.51205C15.0799
9.02605 15.3269 9.66504 15.3079 10.455C15.3048 10.5877 15.3546 10.716
15.4462 10.812C15.5378 10.9079 15.6638 10.9635 15.7964 10.9665C15.929
10.9696 16.0574 10.9198 16.1533 10.8282C16.2492 10.7366 16.3048 10.6107
16.3079 10.478C16.3319 9.44104 15.9979 8.54605 15.3359
7.83205V7.83005C14.6589 7.10405 13.7299 6.72005 12.6119 6.64505L12.5949
6.64305L12.5449 6.64405ZM12.5259 8.31905C12.459 8.31314 12.3916 8.32078
12.3277 8.34151C12.2638 8.36223 12.2047 8.39562 12.154 8.43968C12.1033
8.48374 12.062 8.53757 12.0325 8.59794C12.0031 8.65832 11.9861 8.72401
11.9826 8.79109C11.9791 8.85818 11.9892 8.92528 12.0122 8.98839C12.0352
9.0515 12.0707 9.10932 12.1165 9.15841C12.1624 9.2075 12.2177 9.24685
12.2791 9.2741C12.3405 9.30136 12.4067 9.31596 12.4739 9.31705C12.8919
9.33905 13.1589 9.46504 13.3269 9.63405C13.4959 9.80405 13.6219 10.077
13.6449 10.504C13.6461 10.5711 13.6609 10.6373 13.6882 10.6986C13.7156
10.7599 13.755 10.815 13.8041 10.8607C13.8533 10.9065 13.9111 10.9418

```
13.9742 10.9647C14.0372 10.9876 14.1043 10.9976 14.1713 10.994C14.2383
10.9905 14.3039 10.9735 14.3642 10.944C14.4245 10.9146 14.4783 10.8733
14.5223 10.8226C14.5663 10.7719 14.5997 10.7129 14.6204 10.6491C14.6411
10.5852 14.6487 10.5179 14.6429 10.451C14.6109 9.85105 14.4229 9.32104
14.0379 8.93105C13.6509 8.54105 13.1239 8.35105 12.5259 8.31905Z"
                                        fill="#3C3C3E"></path>
                                </svg>
                                +380 980 178 68</a>
                        </div>
                    </div>
                </section>
            </main>
            </body>
        {% endblock %}
```

```
        from django.shortcuts import render, get_object_or_404, redirect
        from .models import Category, Product, ProductAttribute,
ProductAttributeValue
        from cart.forms import CartAddProductForm
        from django.db.models import Count, Q
        from django.core.paginator import Paginator, EmptyPage,
PageNotAnInteger
        import logging
        from django.http import JsonResponse

        logger = logging.getLogger(__name__)


        def index(request):
            products = Product.objects.all()
            context = {
                'products': products,
            }
            return render(request, "market/index.html", context)


        def delete_product(request, product_id):
```

```
        product = Product.objects.get(id=product_id)
        if request.method == "POST":
            product.delete()
            redirect("/market/")
        context = {'product': product}
        return render(request, "market/deleteproduct.html",
context=context)


    def product_detail(request, product_id):
        product = Product.objects.get(id=product_id)
        form = CartAddProductForm()
        product_attributes =
ProductAttributeValue.objects.filter(product=product)
        return render(request, 'market/detail.html',
                      context={'product': product, 'form': form,
'product_attributes': product_attributes})


    def catalog(request, category_slug=None):
        category = None
        categories = Category.objects.all()
        brands =
Product.objects.values('brand').annotate(count=Count('id')).order_by('b
rand').distinct()
        selected_category_slug = request.GET.get('category')
        selected_brands = request.GET.getlist('brand')
        sort_param = request.GET.get('sort')
        query = request.GET.get('q', '')
        products = Product.objects.filter(available=True)
        if selected_category_slug:
            category = get_object_or_404(Category,
slug=selected_category_slug)
            products = products.filter(category=category)
        if selected_brands:
            products = products.filter(brand__in=selected_brands)
        if sort_param == 'min_price':
            products = products.order_by('price')
        elif sort_param == 'max_price':
```

```
            products = products.order_by('-price')
        if query:
            products = products.filter(Q(title__icontains=query) |
Q(description__icontains=query))
        paginator = Paginator(products, 20)
        page = request.GET.get('page')
        try:
            products = paginator.page(page)
        except PageNotAnInteger:
            products = paginator.page(1)
        except EmptyPage:
            products = paginator.page(paginator.num_pages)
        return render(request, 'market/catalog.html', {
            'category': category,
            'categories': categories,
            'brands': brands,
            'products': products,
            'selected_brands': selected_brands,
            'sort_param': sort_param,
            'query': query,
        })


    def aboutUs(request):
        return render(request, 'market/about_us.html')


    def product_attribute_list(request):
        category_id = request.GET.get('category__id')
        if category_id:
            attributes =
ProductAttribute.objects.filter(category_id=category_id)
            results = [{'id': attr.id, 'text': attr.name} for attr in
attributes]
            return JsonResponse({'results': results})
        return JsonResponse({'results': []})


    from django.db import models
```

```
    from django.utils.text import slugify
    import random
    import string


    def rand_slug():
        return ''.join(random.choice(string.ascii_lowercase +
string.digits) for _ in range(3))


    class Category(models.Model):
        name = models.CharField(max_length=200, db_index=True)
        parent = models.ForeignKey('self', on_delete=models.CASCADE,
related_name='children', null=True, blank=True)
        slug = models.SlugField('URL', max_length=200, unique=True,
db_index=True, null=True, blank=True)
        created_at = models.DateTimeField('Дата створення',
auto_now_add=True)

        class Meta:
            unique_together = (['name', 'parent'])
            verbose_name = 'Категорія'
            verbose_name_plural = 'Категорії'

        def __str__(self):
            full_path = [self.name]
            k = self.parent
            while k is not None:
                full_path.append(k.name)
                k = k.parent
            return '>'.join(full_path[::-1])

        def save(self, *args, **kwargs):
            if not self.slug:
                self.slug = slugify(rand_slug() + '-pickBetter' +
self.name)
            super(Category, self).save(*args, **kwargs)
```

```
    class ProductAttribute(models.Model):
        name = models.CharField(max_length=200)
        category = models.ForeignKey(Category,
related_name='attributes', on_delete=models.CASCADE)

        class Meta:
            verbose_name = 'Характеристика товару'
            verbose_name_plural = 'Характеристики товарів'

        def __str__(self):
            return self.name


    class Product(models.Model):
        category = models.ForeignKey(Category,
related_name='products', on_delete=models.CASCADE, null=True)
        title = models.CharField("назва", max_length=200,
db_index=True)
        slug = models.SlugField("URL", max_length=200, db_index=True,
null=True, blank=True)
        brand = models.CharField("бренд", max_length=200,
db_index=True, null=True, blank=True)
        image = models.ImageField("зображення",
upload_to='products/%Y/%m/%d', blank=True)
        description = models.TextField("опис", null=True, blank=True)
        price = models.DecimalField('ціна', max_digits=10,
decimal_places=2)
        available = models.BooleanField("Наявність", default=True)

        def get_price(self):
            return self.price

        class Meta:
            verbose_name = 'Товар'
            verbose_name_plural = 'Товари'

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

```
    class ProductAttributeValue(models.Model):
        product = models.ForeignKey(Product,
related_name='attribute_values', on_delete=models.CASCADE)
        attribute = models.ForeignKey(ProductAttribute,
related_name='values', on_delete=models.CASCADE)
        value = models.CharField(max_length=200)

        class Meta:
            verbose_name = 'Значення характеристики товару'
            verbose_name_plural = 'Значення характеристик товарів'

        def __str__(self):
            return f'{self.attribute.name}: {self.value}'


    urlpatterns = [
        path('', views.index, name='index'),
        path('about-us', views.aboutUs, name='about_us'),
        path('product/<str:product_id>', views.product_detail,
name="detail"),
        path('catalog/', views.catalog, name='catalog'),
        path('catalog/<str:category_slug>/', views.catalog,
name='catalog_by_category'),
    ]
```

Додаток Б

Код додатку Users

```
from django.db import models
from django.contrib.auth.models import User
from django.db.models.signals import post_save
from django.dispatch import receiver
from cart.models import Cart


class Profile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE,
null=True, blank=True)
    email = models.EmailField(max_length=100, unique=True,
null=True, blank=True)
    contactNumber = models.CharField(max_length=15, default='',
blank=True)
    firstName = models.CharField(max_length=100, blank=True,
null=True)
    lastName = models.CharField(max_length=100, blank=True,
null=True)
    DeliveryAddress = models.CharField(max_length=100, blank=True,
null=True)
    cart = models.OneToOneField(Cart, on_delete=models.SET_NULL,
null=True, blank=True)

    def __str__(self):
        return self.user.username



@receiver(post_save, sender=Profile)
def save_user_profile(sender, instance, **kwargs):
    instance.user.save()


from django.contrib.auth import login, authenticate
from django.contrib.auth.decorators import login_required
```

```
    from django.db.models.signals import post_save
    from django.dispatch import receiver
    from django.shortcuts import render, redirect
    from .forms import RegistrationForm, LoginForm, ProfileUpdateForm,
User
    from order.models import Order
    from cart.models import Cart
    from django.urls import reverse_lazy
    from .forms import CustomPasswordChangeForm
    from django.contrib.auth.views import PasswordChangeView,
PasswordResetView, PasswordResetDoneView, \
        PasswordResetConfirmView, PasswordResetCompleteView
    from .models import Profile


    def register(request):
        if request.method == 'POST':
            form = RegistrationForm(request.POST)
            if form.is_valid():
                user = form.save()
                username = form.cleaned_data['username']
                password = form.cleaned_data['password1']
                user = authenticate(username=username,
password=password)
                if user is not None:
                    login(request, user)
                    return redirect('/')
        else:
            form = RegistrationForm()
        return render(request, 'users/register.html', {'form': form})


    @receiver(post_save, sender=User)
    def create_user_profile(sender, instance, created, **kwargs):
        if created:
            profile = Profile.objects.create(user=instance)
            cart = Cart.objects.create(user=instance)
            profile.cart = cart
            profile.save()
```

```
    def user_login(request):
        if request.method == 'POST':
            form = LoginForm(request.POST)
            if form.is_valid():
                username = form.cleaned_data['username']
                password = form.cleaned_data['password1']
                user = authenticate(request, username=username,
password=password)
                if user is not None:
                    login(request, user)
                    return redirect('/')
        else:
            form = LoginForm()
        return render(request, 'users/login.html', {'form': form})


    @login_required
    def profile(request):
        user = request.user
        profile = user.profile
        active_orders = Order.objects.filter(user=user,

status__in=['Опрацювання', 'Комплектація', 'Доставляється']).order_by(
            '-created_at')
        inactive_orders = Order.objects.filter(user=user,
status__in=['Отримано покупцем', 'Відмінено']).order_by(
            '-created_at')

        context = {
            'user': user,
            'profile': profile,
            'active_orders': active_orders,
            'inactive_orders': inactive_orders,
        }

        return render(request, 'users/profile.html', context=context)
```

```
    def login_required(request):
        return render(request, 'users/login_required.html')


    def update_profile(request):
        if request.method == 'POST':
            form = ProfileUpdateForm(request.POST, request.FILES,
instance=request.user.profile)
            if form.is_valid():
                form.save()
                return redirect('users:profile')
        else:
            form = ProfileUpdateForm(instance=request.user.profile)

        return render(request, 'users/profile_update.html', {'form':
form})


    def order_update_profile(request):
        if request.method == 'POST':
            form = ProfileUpdateForm(request.POST, request.FILES,
instance=request.user.profile)
            if form.is_valid():
                form.save()
                return redirect('order:order-confirmation')
        else:
            form = ProfileUpdateForm(instance=request.user.profile)

        return render(request, 'users/oder_profile_update.html',
{'form': form})


    class CustomPasswordChangeView(PasswordChangeView):
        form_class = CustomPasswordChangeForm
        template_name = 'users/change_password.html'
        success_url = reverse_lazy('')
```

```
    class CustomPasswordResetView(PasswordResetView):
        template_name = 'users/password_reset_form.html'
        email_template_name = 'users/password_reset_email.html'
        success_url = reverse_lazy('users:password_reset_done')


    class CustomPasswordResetDoneView(PasswordResetDoneView):
        template_name = 'users/password_reset_done.html'


    class CustomPasswordResetConfirmView(PasswordResetConfirmView):
        template_name = 'users/password_reset_confirm.html'
        success_url = reverse_lazy('users:password_reset_complete')


    class CustomPasswordResetCompleteView(PasswordResetCompleteView):
        template_name = 'users/password_reset_complete.html'


    from django.urls import path
    from .views import register, profile, update_profile,
order_update_profile, CustomPasswordChangeView, login_required
    from django.contrib.auth.views import LogoutView, LoginView
    from .views import CustomPasswordResetView,
CustomPasswordResetDoneView, CustomPasswordResetConfirmView,
CustomPasswordResetCompleteView


    app_name = "users"

    urlpatterns = [
        path('register/', register, name="register"),
        path('login/',
LoginView.as_view(template_name='users/login.html'), name="login"),
        path('logout/',
LogoutView.as_view(template_name='users/logout.html'), name="logout"),
        path('login_required/', login_required,
name="login_required"),
        path('profile/', profile, name="profile"),
```

```
        path('profile_update', update_profile, name="update-profile"),
        path('order_profile_update', order_update_profile,
name="order-update-profile"),
        path('change_password/', CustomPasswordChangeView.as_view(),
name='change_password'),
        path('password_reset/', CustomPasswordResetView.as_view(),
name="password_reset"),
        path('password_reset/done/',
CustomPasswordResetDoneView.as_view(), name="password_reset_done"),
        path('reset/<uidb64>/<token>/',
CustomPasswordResetConfirmView.as_view(),
name="password_reset_confirm"),
        path('reset/done/', CustomPasswordResetCompleteView.as_view(),
name="password_reset_complete"),
    ]
```

Додаток В

Код додатку Cart

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import login_required
from django.http import JsonResponse
from .models import Cart, CartItem
from market.models import Product
from cart.forms import CartAddProductForm
from order.models import Order, OrderItem
from django.contrib import messages
from django.db import transaction
from django.urls import reverse


@login_required
def add_to_cart(request, product_id):
    product = get_object_or_404(Product, pk=product_id)
    cart, created = Cart.objects.get_or_create(user=request.user)

    if request.method == 'POST':
        form = CartAddProductForm(request.POST)
        if form.is_valid():
            quantity = form.cleaned_data['quantity']
            cart_item, created =
CartItem.objects.get_or_create(cart=cart, product=product)
            if not created:
                cart_item.quantity += quantity
            else:
                cart_item.quantity = quantity
            cart_item.save()
            return redirect('cart:cart-view')
    else:
        form = CartAddProductForm()

    return render(request, 'cart/cart_view.html', {'form': form})
```

```
@login_required
def cart(request):
    cart, created = Cart.objects.get_or_create(user=request.user)
    cart_items = cart.cartitem_set.all()
    total_price = cart.get_total_price()
    context = {
        'cart': cart,
        'cart_items': cart_items,
        'total_price': total_price,
    }
    return render(request, 'cart/cart_view.html', context)


@login_required
def remove_from_cart(request, product_id):
    cart = Cart.objects.get(user=request.user)
    product = Product.objects.get(pk=product_id)
    cart_item = CartItem.objects.get(cart=cart, product=product)
    cart_item.delete()
    return redirect('cart:cart-view')


@login_required
def update_cart(request, cart_item_id):
    cart_item = get_object_or_404(CartItem, pk=cart_item_id)
    if request.method == 'POST':
        action = request.POST.get('action')
        if action == 'increase':
            cart_item.quantity += 1
        elif action == 'decrease' and cart_item.quantity > 1:
            cart_item.quantity -= 1
        elif action == 'decrease' and cart_item.quantity == 1:
            cart_item.delete()
            return redirect('cart:cart-view')
        cart_item.save()
    return redirect('cart:cart-view')


from django.db import models
```

```
    from django.contrib.auth.models import User
    from market.models import Product


    class Cart(models.Model):
        user = models.OneToOneField(User, on_delete=models.CASCADE,
related_name='cart')
        products = models.ManyToManyField(Product, through='CartItem')

        def get_total_price(self):
            total_price = 0
            for item in self.cartitem_set.all():
                total_price += item.product.get_price() *
item.quantity
            return total_price


    class CartItem(models.Model):
        product = models.ForeignKey(Product, on_delete=models.CASCADE)
        cart = models.ForeignKey(Cart, on_delete=models.CASCADE)
        quantity = models.PositiveIntegerField(default=1)

        def get_product_price(self):
            return self.product.price * self.quantity


    from django.urls import path
    from cart import views

    app_name = 'cart'

    urlpatterns = [
        path('add-to-cart/<int:product_id>/', views.add_to_cart,
name='add_to_cart'),
        path('remove/<int:product_id>/', views.remove_from_cart,
name='remove_from_cart'),
        path('', views.cart, name='cart-view'),
        path('update/<int:cart_item_id>/', views.update_cart,
name='update_cart'),
```

```
]
```

Додаток Д

Код додатку Order

```
import json
import hashlib
import requests
from django.http import HttpResponseForbidden, JsonResponse,
HttpResponse
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.decorators import login_required
from market.models import Product
from .models import Order, OrderItem
from cart.models import CartItem, Cart
from django.contrib import messages
from django.db import transaction
from django.urls import reverse
from django.conf import settings
from django.views.decorators.csrf import csrf_exempt
import stripe
from users.models import Profile


def generate_signature(data):
    sorted_string = ';'.join(str(data[k]) for k in
sorted(data.keys()))
    return hashlib.sha1((sorted_string +
settings.WAYFORPAY_SECRET_KEY).encode('utf-8')).hexdigest()


def initiate_payment(order):
    data = {
        'merchantAccount': settings.WAYFORPAY_ACCOUNT,
        'merchantDomainName': 'example.com',
        'orderReference': str(order.id),
        'orderDate': int(order.created_at.timestamp()),
        'amount': str(order.total_price),
        'currency': 'UAH',
```

```
            'productName': [item.product.title for item in
order.orderitem_set.all()],
            'productCount': [str(item.quantity) for item in
order.orderitem_set.all()],
            'productPrice': [str(item.product.price) for item in
order.orderitem_set.all()],
            'language': 'UA'
        }
        data['merchantSignature'] = generate_signature(data)
        response = requests.post(settings.WAYFORPAY_API_URL,
json=data)
        return response.json()


    @login_required
    def process_order(request):
        if request.method == 'POST':
            try:
                cart = Cart.objects.get(user=request.user)
            except Cart.DoesNotExist:
                messages.error(request, "Кошик не знайдено.")
                return redirect('cart:cart-view')

            delivery_address = request.POST.get('delivery_address')
            payment_method = request.POST.get('paymentMethod')

            with transaction.atomic():
                order = Order.objects.create(
                    user=request.user,
                    delivery_address=delivery_address,
                    payment_method=payment_method
                )
                total_price = 0
                for cart_item in cart.cartitem_set.all():
                    order_item = OrderItem.objects.create(
                        order=order,
                        product=cart_item.product,
                        quantity=cart_item.quantity,
                        price=cart_item.product.price *
```

```
cart_item.quantity
                    )
                    total_price += order_item.price

                order.total_price = total_price
                order.save()

                cart.cartitem_set.all().delete()

                if payment_method == 'WayForPay':
                    payment_response = initiate_payment(order)
                    if payment_response.get('error'):
                        messages.error(request, "Помилка оплати: " +
payment_response['error'])
                        return redirect(reverse('order:order-detail',
kwargs={'order_id': order.id}))
                    return redirect(payment_response['payment_url'])
                return redirect(reverse('order:order-detail',
kwargs={'order_id': order.id}))

        else:
            return redirect('cart:cart-view')


    def order_detail(request, order_id):
        order = get_object_or_404(Order, pk=order_id)
        if order.user != request.user:
            return HttpResponseForbidden("You don't have permission to
view this order.")
        return render(request, 'order/order_detail.html', {'order':
order})


    def cancel_order(request, order_id):
        order = get_object_or_404(Order, pk=order_id)
        if request.method == 'POST':
            order.status = Order.CANCELED
            order.save()
            messages.info(request, "Замовлення скасовано.")
```

```
            return redirect('order:order-detail', order_id=order_id)
        return render(request, 'order/cancel_order.html', {'order':
order})


    @csrf_exempt
    def save_transaction_id(request):
        if request.method == 'POST':
            data = json.loads(request.body)
            transaction_id = data.get('transactionId')
            order = Order.objects.filter(id=transaction_id).first()
            if order:
                order.payment_id = transaction_id
                order.save()
                return JsonResponse({'status': 'success',
'transaction_id': transaction_id})
            return JsonResponse({'status': 'error', 'message': 'Order
not found'}, status=404)
        return JsonResponse({'status': 'error'}, status=400)


    def order_confirm(request):
        if not request.user.is_authenticated:
            return redirect('login')  # Перенаправлення на сторінку
логіну, якщо користувач не авторизований

        cart = Cart.objects.filter(user=request.user).first()
        if not cart:
            messages.error(request, "Кошик порожній.")
            return redirect('cart:cart-view')

        cart_items = cart.cartitem_set.all()
        total_price = sum(item.product.price * item.quantity for item
in cart_items)

        context = {
            'cart_items': cart_items,
            'total_price': total_price,
        }
```

```
        return render(request, 'order/confirmation.html', context)


    stripe.api_key = settings.STRIPE_SECRET_KEY
    stripe.api_key = settings.STRIPE_SECRET_KEY


    @login_required
    def confirm_buy_now(request, product_id):
        product = get_object_or_404(Product, pk=product_id)
        profile = Profile.objects.get(user=request.user)

        if request.method == 'POST':
            firstName = request.POST.get('firstName')
            lastName = request.POST.get('lastName')
            contactNumber = request.POST.get('contactNumber')
            email = request.POST.get('email')
            payment_method = request.POST.get('paymentMethod')
            delivery_method = request.POST.get('delivery_method')
            delivery_address = request.POST.get('delivery_address',
''')

            if payment_method == 'creditCard':
                cardNumber = request.POST.get('cardNumber')
                cardExpiry = request.POST.get('cardExpiry')
                cardCVC = request.POST.get('cardCVC')
                if not (cardNumber and cardExpiry and cardCVC):
                    messages.error(request, "Будь ласка, заповніть усі
поля даних карти.")
                    return redirect('order:confirm-buy-now',
product_id=product_id)

            if not (firstName and lastName and contactNumber and
email):
                messages.error(request, "Будь ласка, заповніть усі
обов’язкові поля.")
                return redirect('order:confirm-buy-now',
product_id=product_id)
```

```
            # Створення замовлення
            order = Order.objects.create(
                user=request.user,
                delivery_method=delivery_method,
                payment_method=payment_method,
                delivery_address=delivery_address,
                total_price=product.price
            )
            order_item = OrderItem.objects.create(
                order=order,
                product=product,
                quantity=1,
                price=product.price
            )

            # Створення платіжного наміру Stripe
            if payment_method == 'creditCard':
                intent = stripe.PaymentIntent.create(
                    amount=int(order.total_price * 100),  # Сума в
копійках
                    currency='uah',
                    metadata={'order_id': order.id}
                )
                order.payment_intent_id = intent['id']
                order.save()
                return JsonResponse({'client_secret':
intent['client_secret']})

            order.save()
            messages.success(request, "Ваше замовлення було успішно
створене.")
            return redirect('order:order-success')

        context = {
            'product': product,
            'profile': profile,
        }
        return render(request, 'order/confirm_buy_now.html', context)
```

```
@login_required
def order_success(request):
    return render(request, 'order/success.html')


@csrf_exempt
def stripe_webhook(request):
    payload = request.body
    sig_header = request.META['HTTP_STRIPE_SIGNATURE']
    endpoint_secret = settings.STRIPE_ENDPOINT_SECRET

    try:
        event = stripe.Webhook.construct_event(
            payload, sig_header, endpoint_secret
        )
    except ValueError as e:
        return HttpResponse(status=400)
    except stripe.error.SignatureVerificationError as e:
        return HttpResponse(status=400)

    if event['type'] == 'payment_intent.succeeded':
        payment_intent = event['data']['object']
        order_id = payment_intent['metadata']['order_id']
        order = Order.objects.get(id=order_id)
        order.paid = True
        order.save()

    return HttpResponse(status=200)


from django.db import models
from django.contrib.auth.models import User
from market.models import Product


class Order(models.Model):
    PROCESSING = 'Опрацювання'
    PACKING = 'Комплектація'
```

```
        DELIVERING = 'Доставляється'
        RECEIVED = 'Отримано покупцем'
        CANCELED = 'Відмінено'

        STATUS_CHOICES = [
            (PROCESSING, 'Опрацювання'),
            (PACKING, 'Комплектація'),
            (DELIVERING, 'Доставляється'),
            (RECEIVED, 'Отримано покупцем'),
            (CANCELED, 'Відмінено'),
        ]

        PICKUP = 'Самовивіз'
        DELIVERY = 'Доставка'

        DELIVERY_CHOICES = [
            (PICKUP, 'Самовивіз з магазину'),
            (DELIVERY, 'Доставка'),
        ]

        CASH = 'Готівка'
        CARD = 'Карта'

        PAYMENT_CHOICES = [
            (CASH, 'Готівка'),
            (CARD, 'Карткою'),
        ]

        user = models.ForeignKey(User, on_delete=models.CASCADE)
        products = models.ManyToManyField(Product,
through='OrderItem')
        created_at = models.DateTimeField(auto_now_add=True)
        updated_at = models.DateTimeField(auto_now=True)
        total_price = models.DecimalField(max_digits=10,
decimal_places=2, null=True, default=0)
        status = models.CharField(max_length=20,
choices=STATUS_CHOICES, default=PROCESSING)
        delivery_method = models.CharField(max_length=10,
choices=[('delivery', 'Доставка'), ('pickup', 'Самовивіз')])
```

```
        payment_method = models.CharField(max_length=10,
choices=[('card', 'Кредитна карта'), ('cash', 'Готівка')])
        delivery_address = models.CharField(max_length=100, null=True,
blank=True)
        payment_intent_id = models.CharField(max_length=255,
blank=True, null=True)
        paid = models.BooleanField(default=False)

        def calculate_total_price(self):
            total_price = sum(item.price * item.quantity for item in
self.orderitem_set.all())
            self.total_price = total_price
            self.save()


    class OrderItem(models.Model):
        product = models.ForeignKey(Product, on_delete=models.CASCADE)
        order = models.ForeignKey(Order, on_delete=models.CASCADE)
        quantity = models.PositiveIntegerField(default=1)
        price = models.DecimalField(max_digits=10, decimal_places=2)


    class Payment(models.Model):
        amount = models.DecimalField(max_digits=8, decimal_places=2)
        timestamp = models.DateTimeField(auto_now_add=True)
        success = models.BooleanField(default=False)
        stripe_payment_intent = models.CharField(max_length=255)

        def __str__(self):
            return f"Payment {'success' if self.success else
'failure'} on {self.timestamp.strftime('%Y-%m-%d %H:%M')}"


    from django.urls import path
    from . import views

    app_name = 'order'

    urlpatterns = [
```

```
        path('confirm/', views.order_confirm, name='order-confirm'),
        path('process/', views.process_order, name='process-order'),
        path('detail/<int:order_id>/', views.order_detail,
name='order-detail'),
        path('cancel/<int:order_id>/', views.cancel_order,
name='cancel-order'),
        path('save-transaction/', views.save_transaction_id,
name='save-transaction'),
        path('confirm-buy-now/<int:product_id>/',
views.confirm_buy_now, name='confirm-buy-now'),
        path('webhook/', views.stripe_webhook, name='stripe-webhook'),
        path('success/', views.order_success, name='order-success'),

    ]
```

метадані

Заголовок

Розробка веб-платформи для продажу будівельних матеріалів

Автор
Савчук Володимир

Науковий керівник / Експерт
кандидат технічних наук Олег Пашкевич

підрозділ
King Danylo University

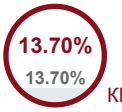
Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

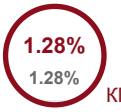
Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	·	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	176

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.



КП 1



КП 2

25
Довжина фрази для коефіцієнта подібності 2

16135
Кількість слів

128856
Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	http://repository.ukd.edu.ua/bitstream/handle/123456789/396/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0%20%D0%A1%D1%82%D1%80%D1%96%D0%BB%D0%B5%D1%86%D1%8C%D0%BA%D0%B8%D0%B9.pdf?sequence=1	47	0.29 %
2	https://www.essuir.sumdu.edu.ua/bitstream/123456789/91891/1/Klishch_bak_rob.pdf	36	0.22 %
3	http://repository.ukd.edu.ua/bitstream/handle/123456789/396/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0%20%D0%A1%D1%82%D1%80%D1%96%D0%BB%D0%B5%D1%86%D1%8C%D0%BA%D0%B8%D0%B9.pdf?sequence=1	33	0.20 %
4	http://repository.ukd.edu.ua/bitstream/handle/123456789/391/%D0%9F%D0%B0%D1%85%D0%BE%D0%BB%D1%8C%D1%87%D1%83%D0%BA%20%D0%9E.%D0%A0.%20%D0%B4%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0.pdf?sequence=1	33	0.20 %