ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка Web-застосунку з продажу техніки з онлайн-системою оплати за допомогою HTML, CSS, Python Django, Angular»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Артур ЛІСЕЦЬКИЙ
(підпис)

Виконав: здобувач вищої освіти групи ПД-42
Артур ЛІСЕЦЬКИЙ

Керівник: Ірина ЗАМРІЙ
д.т.н., доцент

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

_____Лісецькому Артуру Олександровичу_____

1. Тема кваліфікаційної роботи: «Розробка Web-застосунку з продажу техніки з онлайн-системою оплати за допомогою HTML, CSS, Python Django, Angular»
керівник кваліфікаційної роботи д.т.н., доцент, завідувач кафедри ІПЗ Ірина ЗАМРІЙ ,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про методи web-застосунки з продажу техніки онлайн, опис засобів реалізації web-застосунку з продажу техніки онлайн.

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

1. Огляд та аналіз існуючих web-застосунків з продажу техніки онлайн.
2. Проектування web-застосунку з продажу техніки з онлайн-системою оплати.
3. Програмна реалізація та опис функціонування web-застосунку з продажу техніки з онлайн-системою оплати.

4. Тестування web-застосунку.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Схема роботи web-застосунку
6. Діаграма класів.
7. Схема бази даних.
8. Екранні форми.
9. Апробація результатів дослідження

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Огляд існуючих web-застосунків з продажу техніки	14.03.-17.03.2024	
4	Проектування web-застосунку з продажу техніки	18.03-24.03.2024	
5	Розробка web-застосунку	25.03-21.04.2024	
6	Тестування web-застосунку	22.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувач вищої освіти _____ Артур ЛІСЕЦЬКИЙ
(підпис)

Керівник
кваліфікаційної роботи _____ Ірина ЗАМРІЙ
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 63 стор., 1 табл., 56 рис., 10 джерел.

Мета роботи – спрощення процесу продажу техніки онлайн за рахунок застосування web-застосунку, створеного за допомогою HTML, CSS, Python Django, Angular.

Об'єкт дослідження – процес продажу техніки онлайн за допомогою web-застосунків.

Предмет дослідження – автоматизація продажу техніки онлайн за допомогою web-застосунків.

Короткий зміст роботи: В роботі проаналізовано процес продажу техніки онлайн за допомогою web-застосунків. Проаналізовано інструментальні засоби для реалізації web-застосунку з продажу техніки. Розроблено алгоритм роботи застосунку та програмно реалізовані ключові функціональні можливості, зокрема: реєстрація, авторизація користувача, перегляд та редагування профілю користувача, можливість пошуку товарів у каталозі, можливість додавання товарів у кошик, список бажань, можливість оформлення замовлення та перегляд історії замовлень. Проведено функціональне тестування web-застосунку. В роботі використано такі технології, як: HTML, CSS, Python Django, Angular.

Сферою використання web-застосунку є продаж техніки з використанням web-застосунку».

КЛЮЧОВІ СЛОВА: WEB-ЗАСТОСУНОК, PYTHON DJANGO, ANGULAR, HTML, CSS.

ЗМІСТ

ВСТУП

Сфера онлайн-продажу стрімко розвивається, можливість переглядати та замовляти товари онлайн без необхідності відвідування фізичних магазинів надають користувачам багато переваг, таких як: економія часу, більший вибір товарів, можливість придбати необхідний товар з будь-якого місця та пристрою, у якого є доступ до мережі Інтернет. Користувачі мають можливість переглянути відгуки інших покупців про товар.

За результатами дослідження, у 2023 році українці частіше всього замовляли онлайн такі товари, як: електроніка та побутова техніка, тому розробка web-застосунку з продажу техніки з онлайн системою оплати є актуальною та затребуваною темою.

Об’єкт дослідження:

Об’єктом дослідження є процес розробки web-застосунку з продажу техніки з онлайн-системою оплати.

Предмет дослідження:

Вивчення можливостей і особливостей використання технологій HTML, CSS, SCSS, Python Django та Angular для створення функціонального та ефективного електронного магазину з інтегрованою системою оплати.

Мета дослідження:

Розробити повнофункціональний електронний магазин з інтегрованою системою оплати, який буде оптимізований для використання на різних пристроях та забезпечить зручний та безперервний процес покупок для користувачів.

Результати дослідження

У сучасному світі онлайн торгівля дуже стрімко розвивається, і сфера продажу техніки є однією з найбільш затребуваних. Розробка web-застосунку з продажу техніки з онлайн-системою оплати є актуальною, оскільки цей проект дасть змогу людям зручно та безпечно купувати необхідні речі.

Користувачі отримають доступ до широкого асортименту товарів, серед якого вони знайдуть саме те, що їм необхідно за допомогою фільтрів, сортування

та пошуку серед товарів. Сторінки товарів з детальним описом, характеристиками, фотографіями дають можливість користувачу ознайомитись з товаром перед покупкою. Онлайн-система оплати робить процес купівлі товарів у інтернеті легким та безпечним для користувача.

Особистий кабінет користувача дає йому можливість редагувати особисті дані, відстежувати історію замовлень, скасувати замовлення. Система адміністрації web-застосунку забезпечить повний контроль за усіма товарами, представленими у каталозі, а також замовленнями користувачів.

Для створення web-застосунку використовуються сучасні та надійні технології, а саме:

1. HTML – для розмітки структури сайту, контенту сторінок.
2. CSS – для стилізації зовнішнього виду сторінки та елементів.
3. Python Django – для розробки серверної частини web-застосунку.
4. Angular – для створення динамічного та зручного інтерфейсу користувача.

Усі ці технології разом дають змогу створити web-застосунок, який гарантує користувачам надійний захист, зручний інтерфейс та функціонал. Web-застосунок має переваги, серед конкурентів, як для покупців, так і для магазину. Серед переваг для магазину:

1. Збільшення обсягу продажів завдяки зручному інтерфейсу
2. Збільшення конкурентоспроможності
3. Покращення якості досвіду покупок користувачів

Для клієнтів існують такі переваги:

1. Зручність інтерфейсу
2. Широкий вибір товарів
3. Надійний функціонал

1 АНАЛІЗ МЕТОДІВ І ЗАСОБІВ РОЗРОБКИ WEB-ЗАСТОСУНКУ

1.1 Основні завдання web-застосунку

У сучасному світі web-застосунки використовуються у різних сферах життя, в тому числі, у сфері інформаційних технологій, забезпечуючи зручний та інтуїтивно зрозумілий інтерфейс користувача, необхідний функціонал web-застосунку, який дозволяє користувачам отримувати необхідні послуги.

Web-застосунки мають вагомі переваги перед десктопними застосунками, оскільки вони зберігаються на web-серверах та не потребують завантаження та встановлення на пристрій користувача. Це дає можливість користувачам отримати доступ до необхідних застосунків у будь-який час з будь-якого пристрою, у якого є підключення до інтернету.

Основними перевагами web-застосунків є:

1. Крос-платформеність. Користувачі можуть отримати доступ до web-застосунки використовуючи будь-який пристрій, наприклад: телефон, комп’ютер, ноутбук, який має з’єднання з інтернетом.
2. Легка доступність. Web-застосунки не потребують завантаження та встановлення на пристрій користувача.
3. Зручність розробки та підтримки функціональних можливостей web-застосунків.

Оскільки web-застосунки використовуються для різних функцій, наприклад: перегляд відео та фото матеріалів, прослуховування музики, надання послуг онлайн-торгівлі, можливість дистанційного навчання, функціональна частина web-застосунку відрізняється, в залежності від сфери використання. Основними завданнями web-застосунку є:

1. Надання доступ до необхідної інформації
2. Надання необхідних послуг
3. Виконання операцій з обробки даних

4. Забезпечення безпеки даних

У сучасному світі web-застосунки широко використовуються у сфері онлайн-навчання, забезпечуючи можливість перегляду відео уроків, виконання тестів та завдань, створення індивідуального плану навчання для кожного учня, що дає можливість ефективно та зручно навчатись, отримуючи усе необхідне для якісного навчання. Web-застосунки надають доступ до багатьох навчальних матеріалів, книг, статей, що дозволяють отримувати необхідну інформації для навчання.

У сфері здоров'я web-застосунки надають користувачам можливість відслідковувати їх фізичний стан, рівень стресу, якість сну та багато інших показників, необхідних для оцінки фізичного стану користувача. Деякі застосунки дають можливість користувачу зв'язатись з лікарем онлайн для отримання необхідної консультації, без необхідності йти до лікарні.

У сфері онлайн-торгівлі web-застосунки надають користувачу можливість здійснювати пошук, перегляд та порівняння необхідних товарів перед покупкою, не відвідуючи фізичні магазини. Зручний інтерфейс-користувача дає можливість зручно пересуватись сторінками web-застосунку, переглядаючи необхідну інформацію. Системи пошуку та фільтрів забезпечують швидкий та легкий процес пошуку необхідних товарів у web-застосунках. Можливість порівняння товарів за характеристиками, цінами та відгуками інших покупців дають можливість користувачу отримати усю необхідну інформацію про товар перед покупкою. Це дозволяє користувачу полегшити процес вибору та покупки товару, оскільки це можна зробити з будь-якого місця та у будь-який час.

Web-застосунки забезпечують зручний та безпечний процес оплати замовлення онлайн, підтримуючи можливість оплати за допомогою банківських карт, електронних гаманців та платіжних систем.

Після оформлення замовлення користувач має можливість перегляду статусу замовлення та відстеження доставки у режимі реального часу.

Використання web-застосунків у сфері онлайн-торгівлі робить процес покупки для користувачів легшим, зручнішим та безпечнішим.

1.2 Огляд існуючих аналогів

Процес аналізу існуючих аналогів є важливою частиною розробки web-застосунку, оскільки дозволяє визначити сильні та слабкі місця web-застосунків, які вже існують та необхідний функціонал, який потрібен користувачам, для зручного та ефективного використання web-застосунку.

Основними конкурентами web-застосунку з продажу техніки з онлайн-системою оплати на українському ринку є інтернет-магазини Citrus, Moyo та Foxtrot.

1.2.1 Інтернет-магазин Citrus

Citrus [10] – це відомий бренд на українському ринку, який був заснований у 2008 році та спеціалізуються на продажу техніки через мережу Інтернет. Web-застосунок має простий та зрозумілий інтерфейс користувача, який забезпечує зручне використання web-застосунку користувачем.

У web-застосунку присутня можливість здійснювати пошук необхідних товарів не тільки завдяки можливості ввести текстовий запит, а й за рахунок можливості голосового пошуку, дозволяючи користувачу здійснити пошук необхідних товарів за допомогою голосових запитів. Користувач може зручно пересуватись сторінками web-застосунку, використовуючи меню навігації. Користувач має можливість перегляду товару, порівняння його з іншими товарами за різними характеристиками, що дозволяє користувачу обрати необхідний для нього товар. На сайті присутня можливість реєстрації особистого кабінету користувача, який дозволяє відслідковувати оформлені замовлення, слідкувати за статусом замовлення, перегляду історії замовлень.

Присутня можливість додавання товарів у список бажань, який зберігає ці товари та сповіщає користувача, якщо на цей товар з’явились акційні пропозиції.

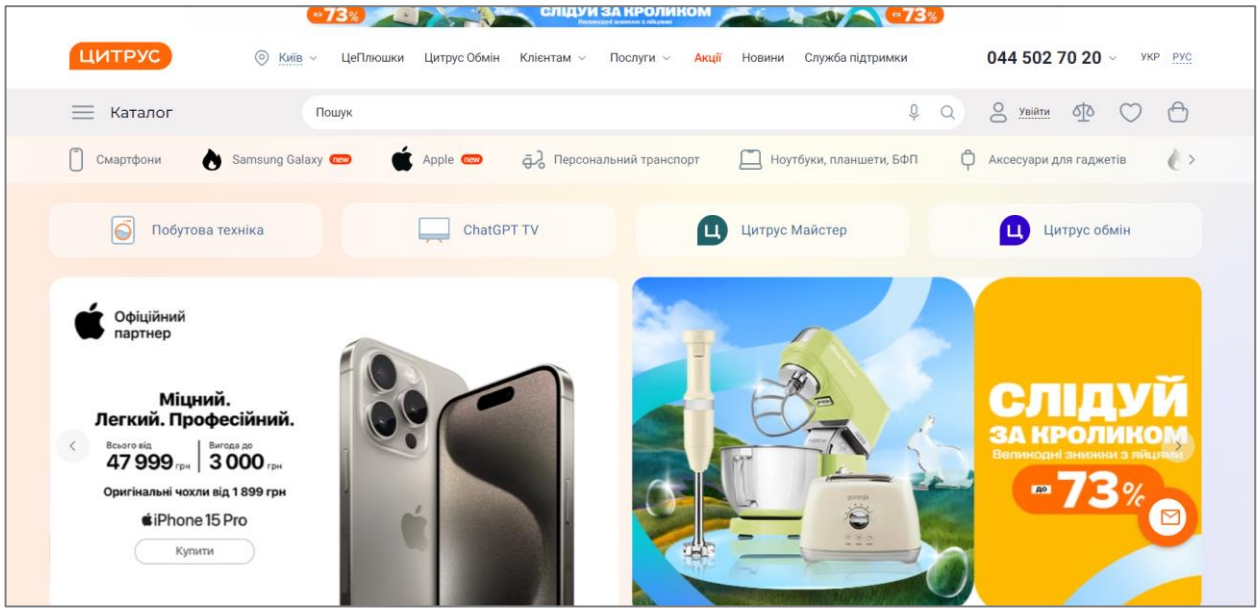


Рис. 1.1 Головна сторінка web-застосунку Citrus.com.ua

При заходженні на сайт, користувач бачить головну сторінку web-застосунку, на якій розміщені каталог товарів, меню пошуку, кнопки для переходу у особистий кабінет користувача, сторінку порівняння товарів, додавання товарів у список бажань та кнопки для переходу у кошик. На головній сторінці web-застосунку розміщенні новини, акції та спеціальні пропозиції для користувачів.

На сайті присутня можливість онлайн-підтримки оператором, який допомагає користувачам вирішувати будь-які проблеми, які можуть виникнути при використанні web-застосунку.

Слабкими сторонами web-застосунку Citrus є відсутня можливість зміни мови інтерфейсу користувача на англійську, оскільки англійська мова є однією із найпопулярніших мов у світі.

1.2.2 Інтернет-магазин Moyo

Moyo [9] – відомий в Україні бренд, який спеціалізуються на продажу техніки та побутової техніки. Інтернет-магазин надає користувачам можливість вибору необхідних товарів серед широкого асортименту товарів.

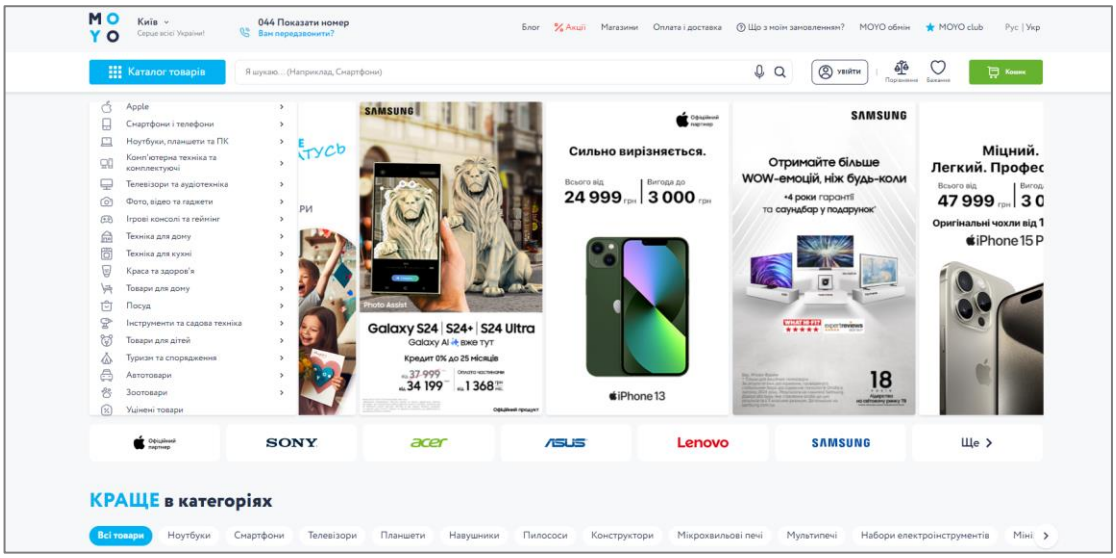


Рис. 1.2 Головна сторінка web-застосунку Moyo.ua

Web-застосунок має простий та зрозумілий інтерфейс користувача, який забезпечує зручне використання web-застосунку користувачем. Присутня система навігації між сторінками web-застосунку, яка дає можливість користувачам зручно пересуватись сторінками web-застосунку.

Каталог товарів містить широкий асортимент, у якого користувач може знайти необхідний товар. Реалізована система пошуку товарів, використовуючи голосові запити, яка дозволяю користувачам здійснювати пошук необхідних товарів не вводячи текстові запити.

Можливість фільтрації та сортування товарів за заданими параметрами дозволяють користувачу швидко переглядати необхідні товари. Присутній особистий кабінет користувача, у якому користувач може переглянути інформації про замовлення, їх актуальний статус та історію попередніх замовлень.

На сайті присутня можливість порівняння товарів між собою, що дає можливість користувачу обрати найкращий для нього товар.

Слабкими місцями Moyo є відсутність можливості перекладу мови інтерфейсу-користувача на англійську.

1.2.3 Інтернет-магазин Foxtrot

Ще одним конкурентом web-застосунку з продажу техніки з онлайн-системою оплати є інтернет-магазин Foxtrot [8], який був заснований у 1994 році та спеціалізуються на продажу техніки, гаджетів та побутової техніки.

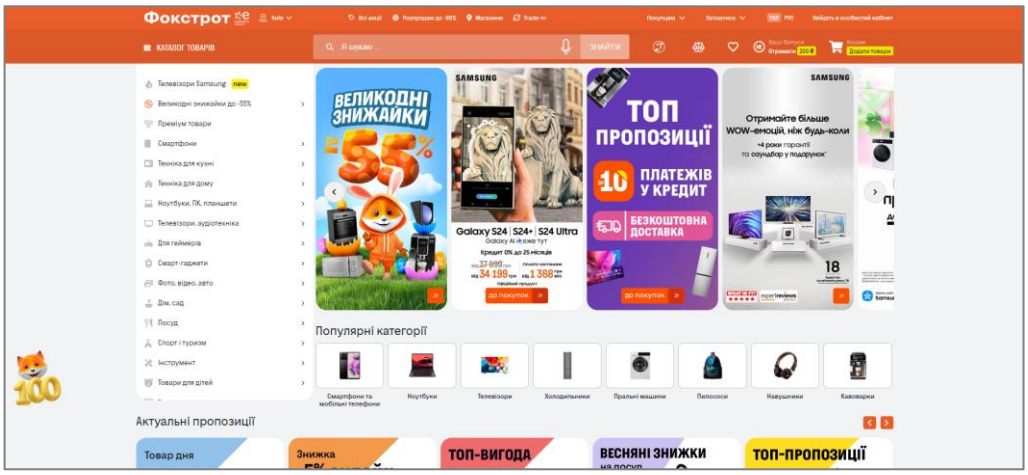


Рис. 1.3 Головна сторінка web-застосунку Foxtrot.com.ua

Web-застосунок має простий та зрозумілий інтерфейс користувача, який забезпечує зручне використання web-застосунку користувачем. Меню навігації дозволяє користувачу зручно пересуватись між сторінками web-застосунку. Реалізована можливість пошуку необхідних товарів за допомогою голосових та текстових запитів, яка дозволяє користувачу швидко здійснювати пошук необхідних товарів.

Web-застосунок має широкий асортимент товарів, який представлений у каталозі, що дозволяє користувачам швидко та зручно переглянути необхідну категорію товарів. Присутня можливість сортування та фільтрації товарів за заданими характеристиками.

Користувач має можливість перегляду інформації про товар та можливість порівняння товару з іншими, що дозволяє отримати усю необхідну інформацію та обрати найкращих товар.

Присутня можливість реєстрації особистого кабінету користувача, у якому можна переглянути актуальну інформацію щодо оформлених замовлень та історію замовлень.

Перевагами Foxtrot є зручна система оплати, яка дозволяє користувачу оплатити замовлення онлайн, використовуючи різні способи оплати, такі як: банківські картки, кредитні картки, оплата за допомогою платіжних систем. На сайті присутня можливість онлайн-підтримки з оператором, який допомагає користувачам у вирішенні проблем, які можуть виникнути при використання web-застосунку.

Слабкими місцями Foxtrot є відсутня можливість зміни мови інтерфейсу користувача та повільна швидкість завантаження сторінок, що спричиняють дискомфорт для користувачів при використання.

Таблиця 1.1

Результати аналізу порівняння аналогів

Характеристика	Foxtrot	Moyo	Citrus	TechoMagic
Мінімальна кількість кроків для замовлення товару	3	4	4	2
Підтримка англійської мови	-	-	-	+
Інтегрована система оплати	+	+	+	+
Адаптивний інтерфейс	+	+	+	+
Швидкість завантаження сторінок	3880 мс	6800 мс	4200 мс	2600 мс

2 МЕТОДИ І ЗАСОБИ РОЗРОБКИ WEB-ЗАСТОСУНКУ

2.1 Вибір засобів та технологій для розробки web-застосунку

Розробка web-застосунку вимагає використання різних засобів та інструментів розробки, які необхідні на різних етапах розробки web-застосунку. Ключовими етапами розробки є Frontend розробка, Backend розробка, створення бази даних.

2.1.1 Frontend розробка

Frontend частина є невід'ємною складовою web-розробки, яка відповідає за створення візуального інтерфейсу web-застосунку. Для реалізації використовується наступні методи та засоби реалізації: HTML, CSS, JavaScript, Angular, React. Ці технології дозволяють створити зручний та ефективний інтерфейс користувачі, який дозволяє легко взаємодіяти з web-застосунком, що значно покращує процес використання web-застосунку користувачем.

HTML [4] – це мови розмітки web-сторінок для браузера, яка забезпечує відображення необхідної інформації на сторінках за допомогою тегів, які вказують браузеру, як саме повинні відображатись різноманітні елементи на сторінках web-застосунку.

CSS – це мова стилів, яка дозволяє визначити візуальне оформлення елементів на сторінках web-застосунку. За допомогою мови стилів можна визначати колір, шрифти, відступи та межі різноманітних елементів для відображення на сторінках. CSS використовується для створення адаптивного дизайну, який коректно відображається на різних пристроях, таких як: телефони, ноутбуки, планшети, комп'ютери та дозволяє користувачам ефективно використовувати web-застосунок на будь-якого з пристроїв.

JavaScript – це мова програмування, яка використовується для задання динамічності елементам, які розміщенні на сторінках web-застосунку. За допомогою неї можна обробляти події користувача на сторінках, виконувати валідацію даних у формах та анімувати різноманітні елементи на сторінках web-застосунку. JavaScript забезпечує необхідний рівень безпеки, оскільки не надає доступу до пам’яті чи процесора, адже була створена тільки бля браузерів, які цього не потребують. Завдяки цьому, небезпечні web-сторінки не мають доступу до конфіденційних даних користувачів та не можуть нанести ніякої шкоди.

Мова програмування JavaScript [6] є найбільш популярною мовою для розробки web-застосунків, оскільки вона легко інтегрується з HTML та CSS та має підтримку усіма сучасними браузерами.

Для створення швидких та якісних інтерфейсів користувача використовуються різні бібліотеки до JavaScript, найбільш популярними з них є React та Angular.

React – це фреймворк до мови програмування JavaScript, який був розроблений компанією Facebook та досить швидко став популярних рішенням серед програмістів з усього світу. Він використовуються для створення інтерфейсів-користувача, забезпечуючи легку масштабованість web-застосунку, у разі необхідності. Ключовою концепцією React є використання компонентів, які є окремими блоками коду, що відповідають за відображення певних частих інтерфейсу користувача на web-сторінках.

Використання React для розробки web-застосунків дає можливість розробити проект будь-якого масштабу за рахунок легкого масштабування, можливості повторного використання компонентів та легкою інтеграцією з іншими бібліотеками та фреймворками. Підтримка серверного рендерингу дозволяє забезпечити швидке завантаження web-сторінок.

Angular – це фреймворк до мови програмування JavaScript, який призначений для створення складних web-застосунків. Використання шаблону проектування Model-View-Controller дозволяє розробляти високо-функціональні web-застосунки

з мінімальною кількістю помилок, завдяки тому, що усі компоненти, необхідні для розробки є організованими та відповідають за окремі частини web-застосунку.

Використання Angular дозволяє розроблювати web-застосунки, роблячи їх організованими та зрозумілими, шо забезпечує швидку та ефективну розробку.

Angular має широкий набір вбудованих наборів інструментів, бібліотек та API, які забезпечують підключення до необхідних при розробці сервісів.

Однією з ключових переваг використання Angular є його структура, яка надає набір правил та сценаріїв, які забезпечують швидку та надійну роботи зв’язків та компонентів.

Для розробки web-застосунку з продажу техніки з онлайн-системою оплати було обрано Angular. Важливим фактором є гарантія підтримки Angular від Google, що надає впевненість у стабільності роботи фреймворку та перспективах розвитку. Angular надає широкий набір інструментів та можливостей для розробки, які включають компонентну архітектуру web-застосунку, вбудовану систему модулів та інструментів для роботи з HTTP-запитами. Це дозволяє ефективно керувати структурою та логікою роботи web-застосунку, що значно спрощує та пришвидшує процес розробки.

2.1.2 Backend розробка

Backend частини web-застосунку відповідає за роботу, пов’язану з функціональними можливостями web-застосунку, забезпечуючи обробку запитів користувача, зберігання, оновлення та редагування даних, можливість реєстрації та авторизації користувача.

Розробка Backend частини потребує глибокого розуміння програмування, різних архітектур системи та засобів безпеки web-застосунків.

Одним з ключових етапів розробки є забезпечення необхідного рівня безпеки web-застосунку. Конфіденційна інформацію користувачів повинна бути надійно захищена від різних типів атак та вразливостей. У сучасному світі інформаційна безпека є однією з ключових проблем, оскільки існує велика кількість вірусів, які

несуть небезпеку для користувачів. Важливо забезпечити надійний захист від різних видів атак, таких як: SQL-ін’єкції, міжсайтові підробки запитів (CSRF), XSS-атаки, які несуть загрозу для користувачів. Для цього необхідно використовувати надійні на перевірені методи захисту, які забезпечать високий рівень захищеності даних та конфіденційної інформації.

Забезпечення можливості масштабування web-застосунку є важливою частиною при розробці. Web-застосунок повинен бути готовий до великого навантаження, зберігаючи можливість забезпечити користувачам доступ до необхідних функціональних можливостей web-застосунку.

2.1.3 Бази даних

Уся інформація про товари та користувачів, необхідна для роботи web-застосунку зберігається у базі даних, яка дає можливість додавання, редагування та видалення інформації, у разі необхідності.

Існують різні типи баз даних, які можуть бути використанні у процесі розробки web-застосунку. Основними з них є:

- Реляційні бази даних, які використовують таблиці для зберігання інформації. Вони працюють за допомогою SQL, забезпечуючи можливість проведення операцій з ними, такі як: додавання, редагування, видалення.
- NoSQL бази даних, які зберігають дані у вигляді документів, ключ значень, графіків та колонок, не використовуючи таблиці. Такий підхід до зберігання даних дозволяє забезпечити можливість масштабування та необхідну продуктивність роботи web-застосунку.

2.2 Вибір середовища розробки

Для розробки web-застосунку з продажу техніки з онлайн-системою оплати було обрано наступні середовища програмування: Visual Studio Code, JetBrains WebStorm, JetBrains Pycharm.

2.2.1 Visual Studio Code

Visual Studio Code – це текстовий редактор коду, який був розроблений у 2015 році компанією Microsoft. Редактор має підтримки налагодження, підсвічування синтаксису та вбудований контролер версій Git. Visual Studio Code має підтримки більшості сучасних мов програмування, таких як: C, C#, JavaScript, Python та є зручним для роботи з HTML та CSS файлами.

Цей редактор ідеально підходить для написання верстки сторінок web-застосунку, оскільки забезпечує широкий набір можливостей, які пришвидшують процес розробки, при цьому, не створюючи навантаження на систему, що дозволяє швидко та ефективно працювати.

Visual Studio Code має зручний та зрозумілий інтерфейс користувача, який дозволяє зручно пересуватись необхідними вкладками, не витрачаючи час на пошук необхідних елементів.

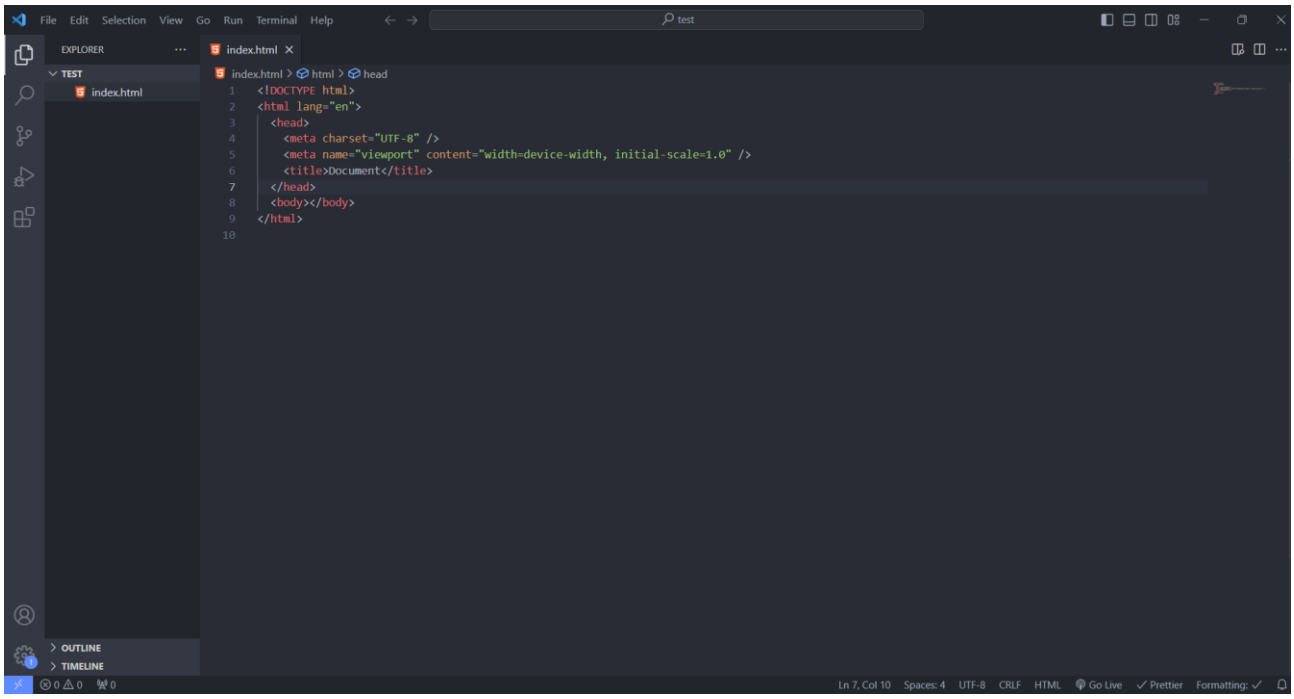


Рис. 2.1 Інтерфейс Visual Studio Code

Присутня можливість доповнювати необхідний функціонал редактору за допомогою безкоштовних розширень, які доступні у вбудованій бібліотеці розширень.

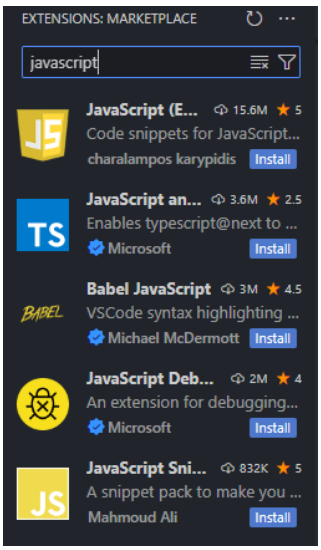


Рис. 2.2 Бібліотека розширень Visual Studio Code

Редактор має вбудовані засоби для відлагодження коду, які дозволяють виявляти та виправляти помилки. Це полегшує процес перевірки коду та виявлення проблем.

2.2.2 JetBrains WebStorm

JetBrains WebStorm – це інтегроване середовище розробки для JavaScript, HTML та CSS. Він підтримує наступні мови програмування: JavaScript, TypeScript та Dart. Редактор має функції автодоповнення, аналізу коду, рефакторингу та інтеграцію з системою управління версіями Git.

Редактор використовується для написання більш складних проектів, оскільки він підтримуєте роботи з фреймворками React та Angular, необхідних для розробки якісного та зручного інтерфейсу користувача.

JetBrains WebStorm має інтеграції з системами відслідковування помилок – це програми, створені з метою враховувати та контролювати помилки, знайдені у

web-застосунку. Також є можливість відслідковувати процес виправлення цих помилок. Така система є необхідним компонентом для розробки якісного web-застосунку.

Редактор має можливості віддаленого розгортання за протоколами FTP, SFTP з можливістю автоматичної синхронізації файлів між локальним комп'ютером та сервером.

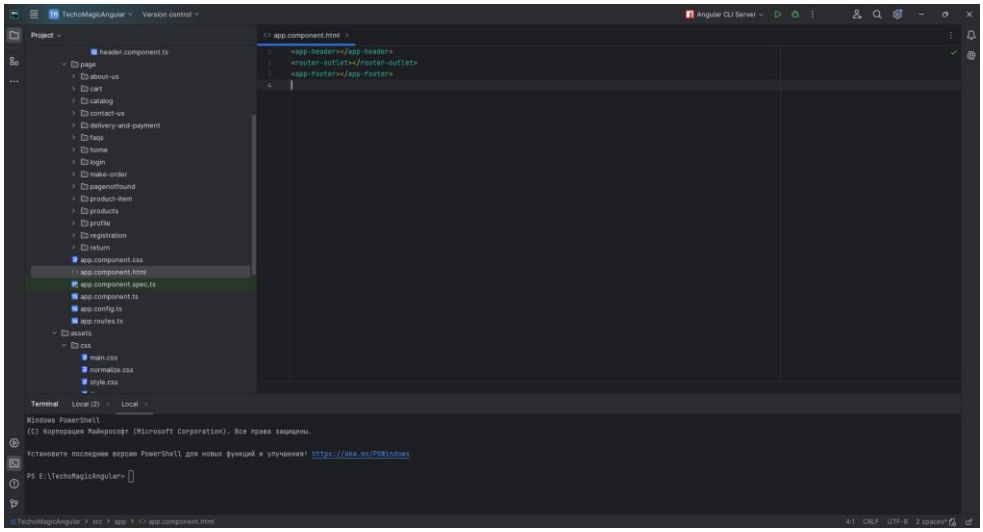


Рис. 2.3 Інтерфейс JetBrains WebStorm

JetBrains PyCharm – це інтегроване середовище розробки для мови програмування Python. Редактор надає засоби для аналізу коду та інструмент для написання юніт-тестів для тестування. Головною перевагою PyCharm, у порівнянні з іншими середовищами розробки, є підтримка розробки за допомогою фреймворку Django, необхідного для розробки Backend частини web-застосунку.

JetBrains PyCharm має інтегрований відлагоджувач з багатьма корисними функціями, такі як breakpoints, перегляд значень змінних, стек викликів. Це забезпечує якісну розробку та тестування web-застосунку.

Редактор має підтримку різних систем контролю версій, що дозволяє відслідковувати зміни у розробці web-застосунку.

Система контролю версій – це програмне забезпечення, що дозволяє відслідковувати зміни у файлах при розробці web-застосунку. Система зберігає

копії кожної версії файлу, що дозволяє повернутись до попередніх версій у разі необхідності.

Також є можливість декільком людям працювати одночасно над одним проектом, об’єднуючи зміни, які вносять розробники.

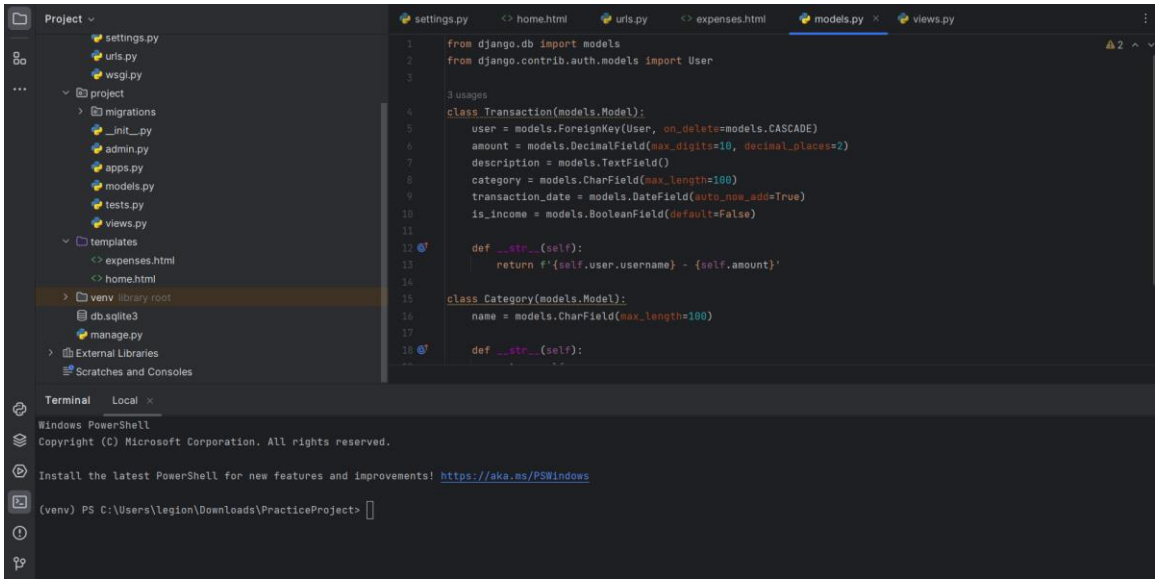


Рис. 2.4 Інтерфейс JetBrains PyCharm

2.3 Вибір засобів розробки web-застосунку

Для реалізації web-застосунку з продажу техніки з онлайн-системою оплати було обрану технології HTML, CSS, JavaSript, Angular та Python Django.

2.3.1 HTML

HTML – це мова розмітки для web-сторінок, яка визначає вміст і структуру елементів, які відображаються на сторінках web-застосунку. Браузер отримує HTML-файли та передає їх вміст на сторінки web-застосунку, забезпечуючи відображення необхідних елементів.

Кожен елемент є блоком, у який можуть бути вбудовані зображення, форми, тексти, посилання та інші елементи. За допомогою HTML створюється семантична

та структурована сторінка, яка має заголовки, абзаци, списки, таблиці та посилання. Елементи відображаються за допомогою використання відповідних тегів, за допомогою яких браузер визначає як саме повинен відображатись елемент на сторінці.

2.3.2 CSS

CSS – це мова стилів, яка дозволяє стилізувати елементи, які відображаються на сторінках web-застосунку. За допомогою CSS визначається колір, відступи, положення елементів на сторінках. Також за допомогою одного CSS файлу можна керувати налаштуваннями елементів на різних сторінках web-застосунку. Це дозволяє зменшити кількість коду та файлів, необхідних для роботи web-застосунку, що робить процес розробки більш організованим та зрозумілим.

Крім того, за допомогою мови стилів CSS можна адаптувати візуальну частину web-застосунку під різни пристрої, такі як: телефони, планшети, ноутбуки, комп’ютери. Це дозволяє користувачам використовувати web-застосунок з будь-якого пристрою, який має підключення до мережі інтернет.

2.3.3 JavaScript

JavaScript – це мова програмування, яка дозволяє реалізовувати інтерактивні елементи на сторінках web-застосунку, використовуючи різні функції. Можливість взаємодіяти з користувачем за допомогою інтерактивних елементів, можливість керування браузером, обміном даними між користувачем та сервером реалізується за допомогою JavaScript.

Для того, щоб користувач між взаємодіяти з елементами на web-сторінках, використовуються JavaScript скрипти, які запускаються на пристрої користувача та дають можливість взаємодіяти з елементами на web-сторінках. JavaScript має вбудовані об’єкти та функції, які дозволяють працювати з даними та елементами на сторінках web-застосунку.

JavaScript має декілька важливих переваг, які дозволяють розробникам реалізовувати необхідний функціонал web-застосунку, серед них:

1. Простота вивчення. JavaScript має зрозумілий та простий синтаксис, який дозволяє розробникам швидко приступити до реалізації необхідного функціоналу web-застосунку.
2. Широкий вибір бібліотек та фреймворків. За допомогою широкого набору різноманітних бібліотек та фреймворків розробники мають можливість реалізувати необхідний функціонал web-застосунку.
3. Універсальність. JavaScript може використовуватись як на клієнтській стороні web-застосунку, так і на серверній частині, що дозволяє розробникам використовувати одну мову програмування для реалізації різних завдань.

2.3.4 Angular

Для розробки якісного інтерфейсу користувача використовуються фреймворк до мови програмування JavaScript Angular. Angular – це фреймворк, написаний на мові програмування TypeScript [2], який відіграє ключову роль у розробці Frontend частини web-застосунку. Забезпечуючи необхідну стабільність та ефективну роботу web-застосунку, Angular є надійним рішенням для реалізації web-застосунку.

Angular дає можливість створювати Single Page Applications – це додатки, які можуть працювати без необхідності перезавантаження сторінок для оновлення інформації на сторінках. Основними можливостями та перевагами використання Angular є:

1. Компонентна архітектура, завдяки якій, незалежні блоки коду можна використовувати повторно, зберігаючи чистоту та читабельність коду.
2. Двостороннє зв’язування даних, що забезпечує постійний зв’язок між шаблонами та компонентами. Це дає змогу динамічно оновлювати необхідні дані у web-застосунку.

3. Маршрутизація, яка пов’язує компоненти web-застосунків з URL посиланнями, що забезпечує переходи на різні сторінки web-застосунку без перезавантаження web-сторінки.

Angular використовує компонентно-орієнтований підхід розробки web-застосунків, завдяки чому уся необхідна функціональна частина web-застосунків зберігається у компонентах, кожен з яких відповідає за окрему частину інтерфейсу сторінки web-застосунку. Кожен компонент містить свої властивості, методи та шаблони, які взаємодіють між собою завдяки двосторонньому зв’язуванню даних, що дозволяє автоматично оновлювати інтерфейс користувача у разі необхідності.

Використання мови програмування TypeScript покращує читабельність коду, при цьому зменшуючи кількість помилок при розробці web-застосунку.

Перевагами використання Angular [1] є зручний спосіб організації коду завдяки компонентам, які зберігають окремі функціональні частини інтерфейсу. Це значно полегшує процес документування та покращує чистоту та читабельність коду.

Angular має широкий вбудований набір інструментів, які забезпечують зручний та швидкий процес створення компонентів та модулів, необхідних для роботи web-застосунку. Angular постійно оновлюється та підтримується, це гарантує необхідну стабільність роботи web-застосунку та його довгий термін підтримки.

2.3.5 Python

Для розробки Backend частини web-застосунку було обрано мову програмування Python та фреймворк Django [7], який дозволяє швидко та якісно розробити необхідний функціонал для роботи web-застосунку.

Python – це одна з найпопулярніших мов програмування, яка використовується для реалізації різних web-застосунків. Завдяки зрозумілому синтаксису, гарною читабельністю коду та широкий спільноті розробників, Python є ефективним інструментом для розробки web-застосунків. Широкий набір

бібліотек та фреймворків дозволяє реалізовувати необхідний функціонал, необхідний для роботи web-застосунку.

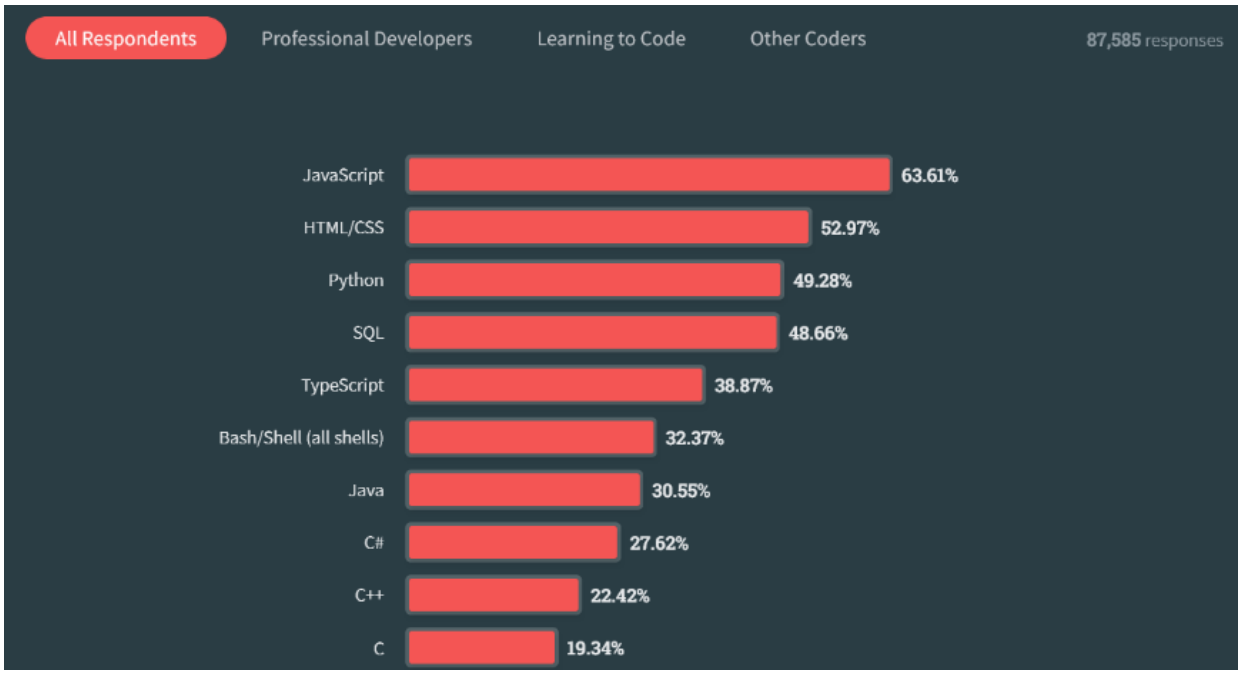


Рис. 2.5 Найпоширеніші мови програмування у 2023 році

Перевагами використання Python Django є швидкість розробки, яка досягається за рахунок використання модулів та компонентів, які є вбудованими у фреймворк. Для керування базою даних, процесами реєстраціє та авторизації, управління сесіями використовується вбудована панель для адміністрації web-застосунку.

Python Django має підтримки маршрутизації URL, які дозволяє користувачам швидко переміщуватись між сторінками web-застосунку. Django постійно підтримується та оновлюється, забезпечуючи необхідний рівень надійності для розробки web-застосунків.

Важливою частиною розробки web-застосунку є забезпечення безпеки даних, для цього використовуються засоби безпеки, які вбудовані у Django. Вони захищають web-застосунок від різних видів атак, таких як: SQL-ін’єкції, міжсайтові сценарії XSS, підробки міжсайтових запитів.

Система авторизації користувача забезпечує необхідний рівень безпеки, що робить процес керування особистими даними користувача безпечним.

2.4 Вибір СУБД для управління даними

Для зберігання необхідної для роботи web-застосунку інформації необхідно використовувати базу даних. Для управління базами даних існує велика кількість різних систем, серед яких було обрано SQLite, оскільки вона є простою у використанні та ідеально підходить для зберігання інформації.

SQLite не потребує для роботи окремого серверу, це робить її зручним у використання для роботи web-застосунку. Уся необхідна інформацію зберігається в одному файлі, що значно спрощує процес її розгортання та підтримки, завдяки чому система стає дуже мобільною.

Завдяки підтримці великої кількості функціональних можливостей, SQLite забезпечує необхідний функціонал для роботи з базою даних, а саме: індекси, зовнішні ключі, унікальні ключі та підзапити.

Однією з головний переваг використання SQLite є те, що для її роботи не потрібно встановлювати додаткове програмне забезпечення, це спрощує процес розробки та дозволяє зменшити час, необхідний для налаштування.

SQLite підтримує більшість популярних мов програмування, що робить її універсальним інструментом для різних видів web-застосунків.

3 РЕАЛІЗАЦІЯ WEB-ЗАСТОСУНКУ

3.1 Розробка архітектури системи

Розробка архітектури web-застосунку є важливою частиною розробки web-застосунку, оскільки вона визначає структуру системи, взаємодію її компонентів між собою, а також технології, які використовуються для розробки web-застосунку.

Клієнтська сторона web-застосунку відповідає за відображення інтерфейсу користувача, доступних елементів на сторінках web-застосунку. Клієнтська сторона включає в себе сторінки web-застосунку, такі як: головна сторінка, сторінка логіну та реєстрації, сторінка каталогу, сторінка перегляду товару, сторінка особистого кабінету, сторінка оформлення замовлення. Також відображення кнопок для реєстрації, авторизації, кнопка переходу у кошик, особистий кабінет користувача.

Клієнтська сторона виконує функцію валідації даних перед тим, як вони будуть відправлені на сервер для опрацювання та відповідає за відображення даних, які отримані у відповідь.

Серверна сторона web-застосунку відповідає за оброблення даних, отриманих від клієнтської частини web-застосунку, взаємодіє з базою даних, дозволяючи додавати, редагувати та видаляти інформацію.

База даних зберігає особисті дані користувачів та виконує запити від серверної частини web-застосунку на додавання, редагування та видалення інформації.

Було розроблено діаграму використання web-застосунку, яка використовується для моделювання функціональних можливостей користувачів у системі. Діаграма містить дії, які доступні користувачу. Для відображення елементів на діаграмі використовуються актори, сценарії використання та залежності між ними.

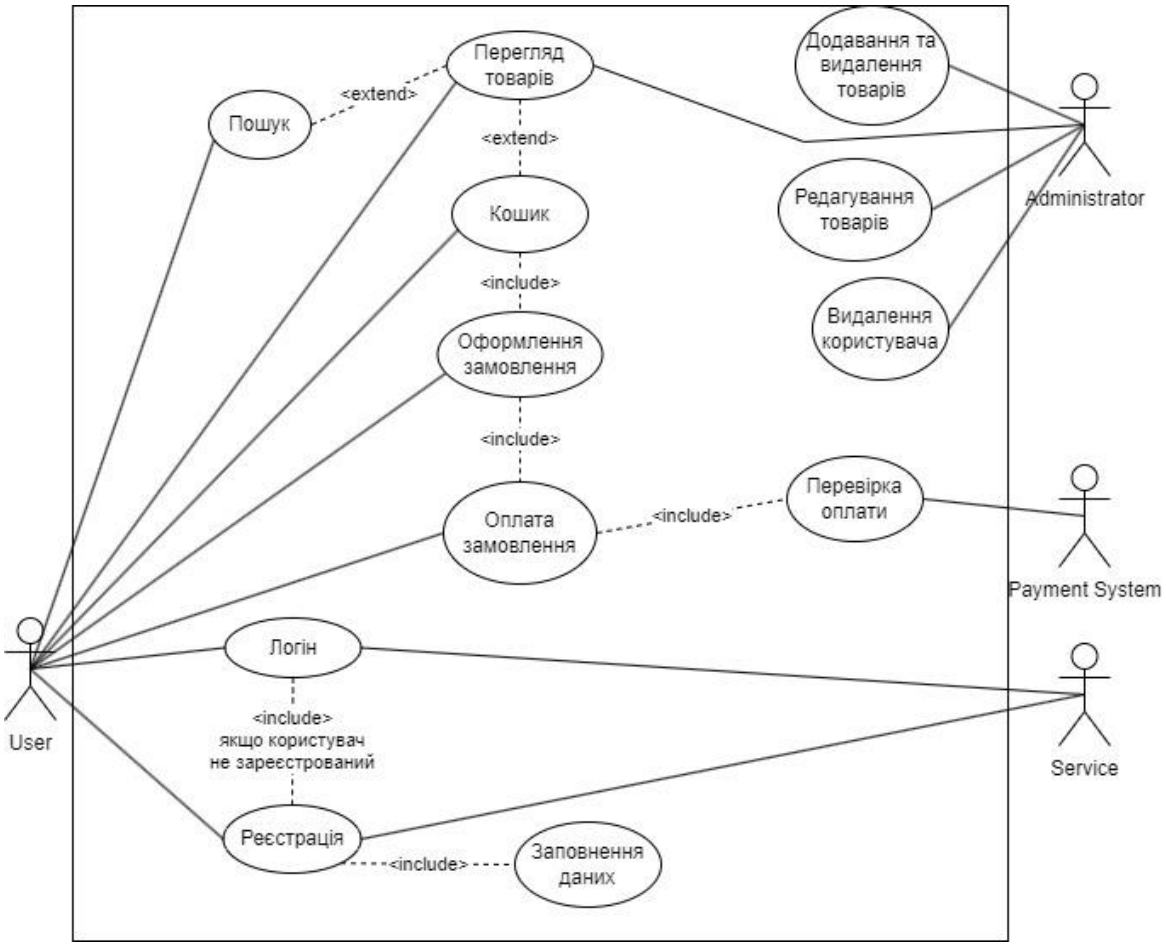


Рис. 3.1 Діаграма варіантів використання

Користувач має можливість зареєструватись, або увійти до особистого кабінету користувача, якщо він вже зареєстрований. Користувач може здійснювати пошук товарів завдяки каталогу товарів, переглядати товари, додавати товари у кошик, здійснити оформлення та оплату замовлення. Оплата замовлення включає перевірку оплати. Адміністратор може переглядати усі товари, редагувати їх. Також присутня можливість для адміністратору додавати, видаляти товари та видаляти користувачів.

Діаграма класів є важливою частиною опису структури роботи web-застосунку, оскільки представляє різні частини web-застосунку та їх взаємодія між собою.

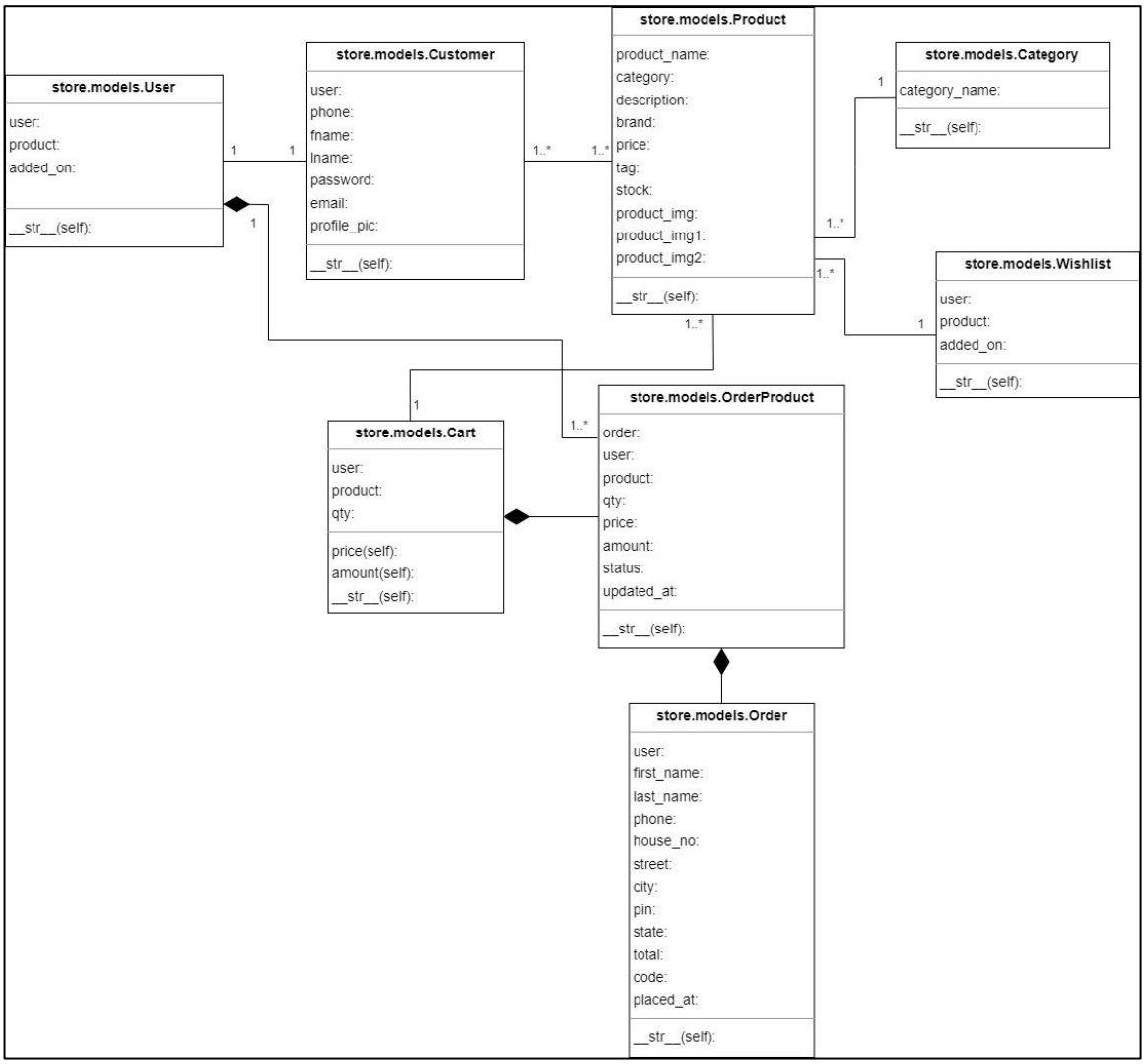


Рис. 3.2 Діаграма класів

Було розроблено схему роботи web-застосунку, яка відображає, як саме буде працювати web-застосунок. Клієнтська частина web-застосунку відповідає за відображення елементів на сторінках. Вона надсилає запити до серверної частини, отримуючи у відповідь дані для відображення.

Серверна частина web-застосунку надсилає SQL запити у базу даних, отримуючи у відповідь дані з бази даних, які передає у клієнтську частину. Платіжна система відповідає за проведення оплати, отримуючи від клієнтської частини web-застосунку запити на оплату. У відповідь платіжна система надсилає статус оплати.

Рис. 3.3 Схема роботи web-застосунку

3.2 Проектування макету web-застосунку

Проектування макету web-застосунку є важливою частину процесу розробки web-застосунку. Важливо розробити візуальну концепцію та структуру web-застосунку, яка відображає повну функціональність web-застосунку. Від правильно спроектованого макету повністю залежить візуальна складова частину web-застосунку, зручність його користування, ефективність взаємодії з web-застосунком. Під час проектування важливо враховувати потреби потенційних клієнтів web-застосунку, їх вимоги та побажання. Це забезпечить високий рівень комфортну та задоволення для клієнтів під час використання web-застосунку.

Крім того, важливо дотримуватись принципів дизайну, щоб ефективно впровадити усі необхідні для роботи web-застосунку елементи на web-сторінках. Необхідно забезпечити високий рівень зрозумілості, для того, щоб користувачу було зручно та інтуїтивно зрозуміли переміщуватись між сторінками web-застосунку. Проектування макету також включає в себе врахування адаптивності

web-застосунку, щоб користувачі на різних пристроях, таких як планшет, телефон, комп’ютер, ноутбук, могли зручно використовувати web-застосунок.

Основними принципами для розробки якісного web-застосунку є:

1. Зручність використання
2. Адаптивність
3. Швидкість завантаження
4. Кольори та типографіка

Одразу після відкриття web-застосунку користувач повинен зрозуміти, як орієнтуватись у web-застосунку. Для цього, необхідно розробити простий та зрозумілий користувачам інтерфейс. Важливо дотримуватись одного стилю на всіх сторінках web-застосунку, розміри та види кнопок, шрифти, розташування картинок. Усе це має бути одноманітним на усіх сторінках для комфортного користування web-застосунком.

Адаптивність є важливою частиною при проектуванні інтерфейсу web-застосунку, оскільки користувачі можуть використовувати різні пристрої під час використання web-застосунку, необхідно забезпечити коректне відображення усіх елементів на web-сторінках на усіх пристроях. Якщо адаптивність не реалізована, або реалізована не достатньо якісно, користувачі, які використовують мобільні телефони для користуванням web-застосунку просто не зможуть виконати необхідні дії на web-сторінках.

Швидкість завантаження web-сторінок є важливою складовою, яку необхідно враховувати при створені інтерфейсу для web-застосунку, оскільки саме швидкість завантаження сторінок першочергово впливає на сприйняття користувачем web-застосунку. Швидке завантаження сторінок створює враження про ефективність та професіоналізм, тоді як повільна швидкість завантаження може викликати негативні емоції у користувачів.

Також швидкість завантаження сторінок надзвичайно важливо для пошукової оптимізації, оскільки саме час завантаження web-сторінок є важливою складовою під час ранжування результатів пошуку. Таким чином, повільна

швидкість завантаження сторінок може негативно вплинути на позиції у результатах пошукових систем.

Для розробки інтерфейсу web-застосунку можна використовувати багато різних програм та сервісів, проте, найпопулярнішою з них є Figma.

Figma – це онлайн-сервіс для розробки макетів інтерфейсу для сайтів та web-застосунків. Сервіс дозволяє користувачам взаємодіяти у режимі реального часу для сумісного створення та редагування макетів інтерфейсів. Користувачі можуть одночасно вносити зміни до проекту, відслідковуючи зміну у режимі реального часу, це робить процес розробки макету інтерфейсу більш швидким та зручним.

Перевагою використання Figma [5] є те, що сервіс містить багато інструментів для дизайну, таких як, векторні малюнки, форми, тексти, шрифти, іконки. Усе це дозволяє розробникам створити якісний та зрозумілий користувачам інтерфейс web-застосунку.

Figma має велику кількість різноманітних плагінів, які розширюють функціонал сервісу та дозволяють інтегрувати його з іншими інструментами.

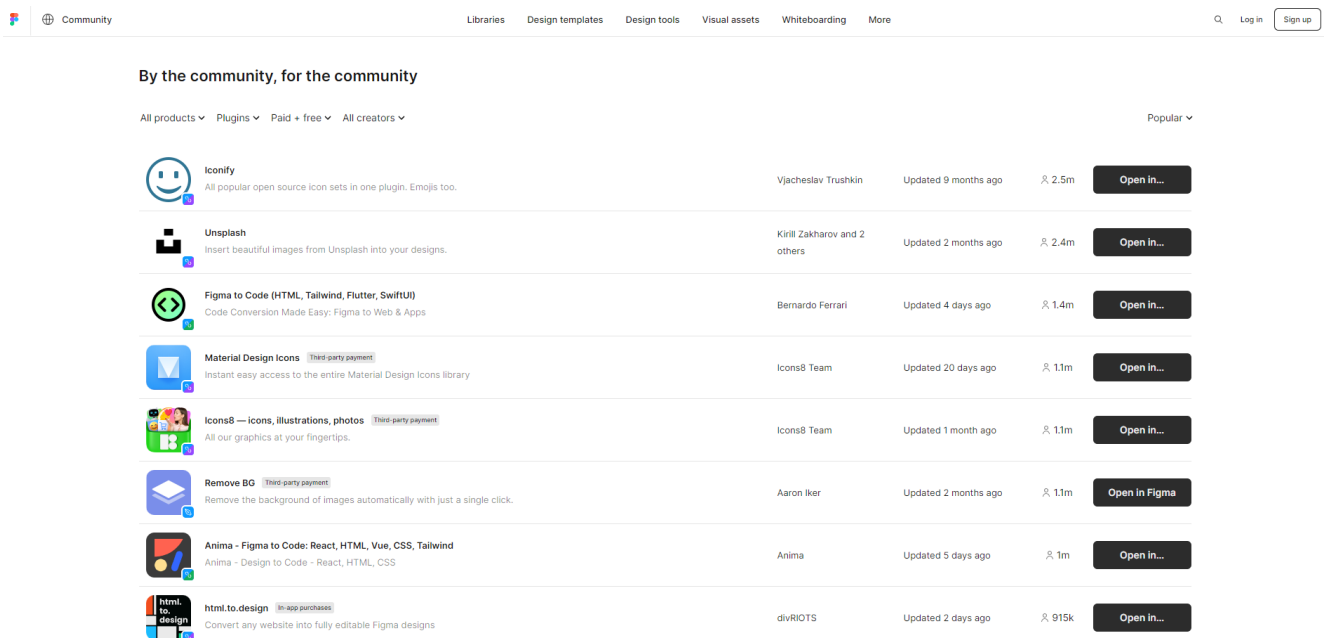


Рис. 3.4 Плагіни для Figma

Макети сторінок повинні містити:

1. Header web-застосунку, у якому відображаються логотип, меню навігації, кнопки для авторизації та реєстрації, перемикач світлої та темної теми, кнопку кошику.
2. Footer web-застосунку, у якому відображаються логотип, меню навігації, кнопки для авторизації та реєстрації.
3. Головний зміст сторінки, який відображає основний контент.

За допомогою сервісу Figma було створено макети сторінок web-застосунку, які будуть використовуватись для верстки та реалізації інтерфейсу користувача.

Головна сторінка web-застосунку повинна бути простою та зрозумілою, для того, щоб у користувача не виникло проблем з користуванням та навігацією сторінками web-застосунку. Було розроблено навігаційне меню, яке має однаковий вигляд на всіх сторінках web-застосунку та дозволяє користувачу пересуватись між сторінками.

Header web-застосунку містить наступні елементи:

- Логотип, який виконує функцію переходу на головну сторінку web-застосунку.
- Кнопка «Про нас», яка дозволяє користувачу перейти на сторінку з інформацією про web-застосунок.
- Кнопка «Контакти», яка дозволяє користувачу перейти на сторінку контактів, що містить інформацію та контакти для зв’язку.
- Кнопка «Каталог», яка дозволяє користувачу перейти на сторінку каталогу.
- Кнопка «Логін», яка дозволяє користувачу перейти на сторінку логіну для входу у особистий кабінет користувача.
- Кнопка «Реєстрація», яка дозволяє користувачу перейти на сторінку реєстрації для створення особистого кабінету користувача.
- Кнопка «Кошик», яка дозволяє перейти на сторінку кошику.
- Перемикач для зміни світлої та темної теми.

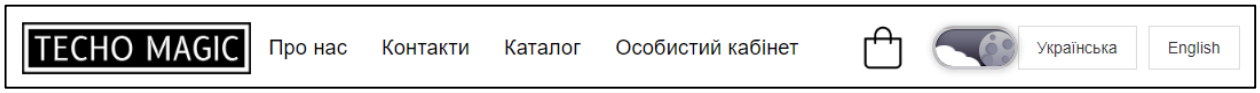


Рис. 3.5 Макет меню навігації

Footer web-застосунку має однаковий вигляд на всіх сторінках web-застосунку та містить наступні елементи:

- Логотип, який виконує функцію переходу на головну сторінку web-застосунку.
- Кнопка «Контакти», яка дозволяє користувачу перейти на сторінку контактів, що містить інформацію та контакти для зв’язку.
- Кнопка «Каталог», яка дозволяє користувачу перейти на сторінку каталогу.
- Кнопка «FAQs», яка дозволяє користувачу перейти на сторінку з відповідями на популярні запитання.
- Кнопка «Доставка і оплата», яка дозволяє користувачу перейти на сторінку з інформацією про доставку і оплату.
- Кнопка «Умови повернення», яка дозволяє користувачу перейти на сторінку з інформацією про умови повернення товарів.
- Кнопка «Логін», яка дозволяє користувачу перейти на сторінку логіну для входу у особистий кабінет користувача.
- Кнопка «Реєстрація», яка дозволяє користувачу перейти на сторінку реєстрації для створення особистого кабінету користувача.

Рис. 3.6 Макет футеру web-застосунку

Основна частина головної сторінки містить перелік популярних товарів, представлених у web-застосунку, натиснувши на які, користувач переходить на сторінку товару, перелік логотипів брендів, товари яких представлені у web-застосунку

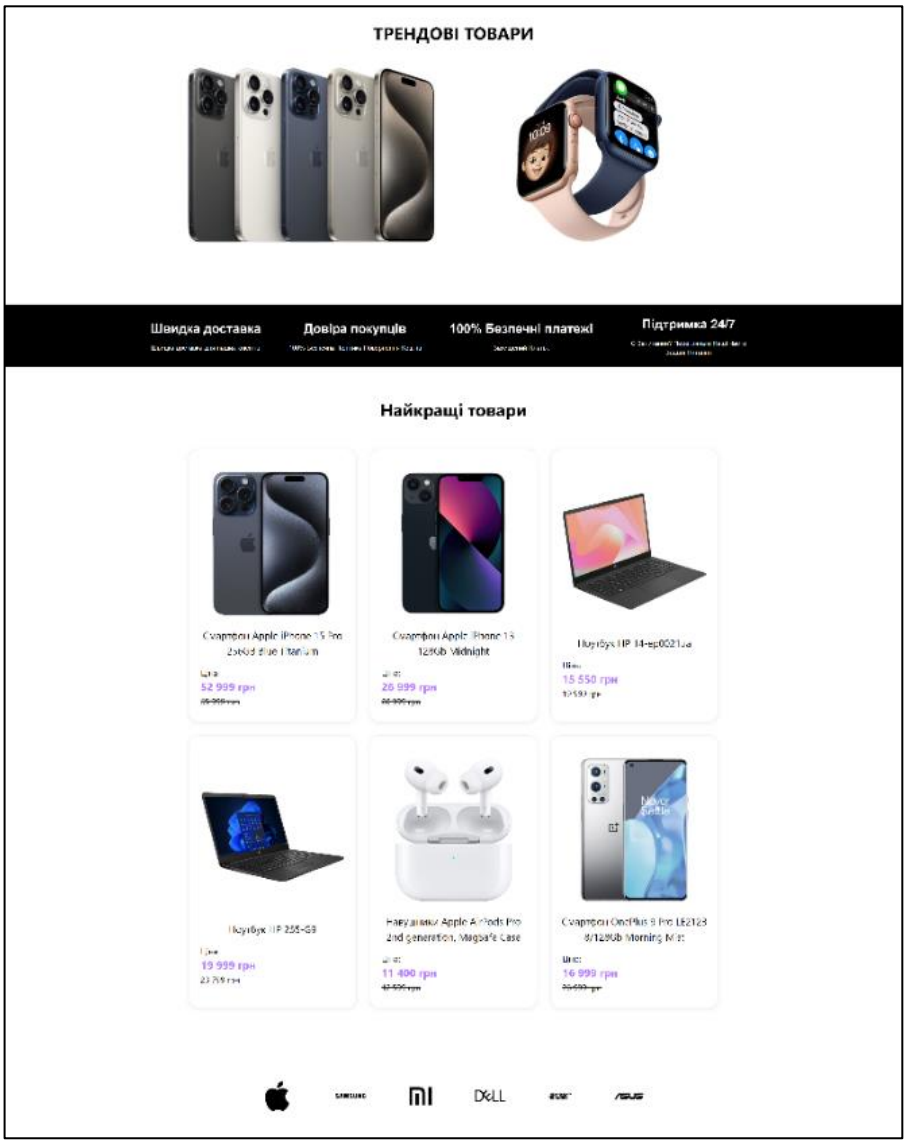


Рис. 3.7 Макет головної сторінки web-застосунку

На сторінці «Про нас» були розміщені текстові елементи, які містять загальну інформацію про web-застосунок, що дозволить користувачу дізнатись більше необхідної інформацію.

Рис. 3.8 Макет сторінки «Про нас»

На сторінці «Контакти» були розміщені текстові елементи, які містять актуальні дані для зв’язку, за якими користувач може зв’язатись з представниками web-застосунку, у разі виникнення запитань.

Також була розміщена форма зворотного зв’язку, яка дозволяє користувачам надіслати повідомлення.

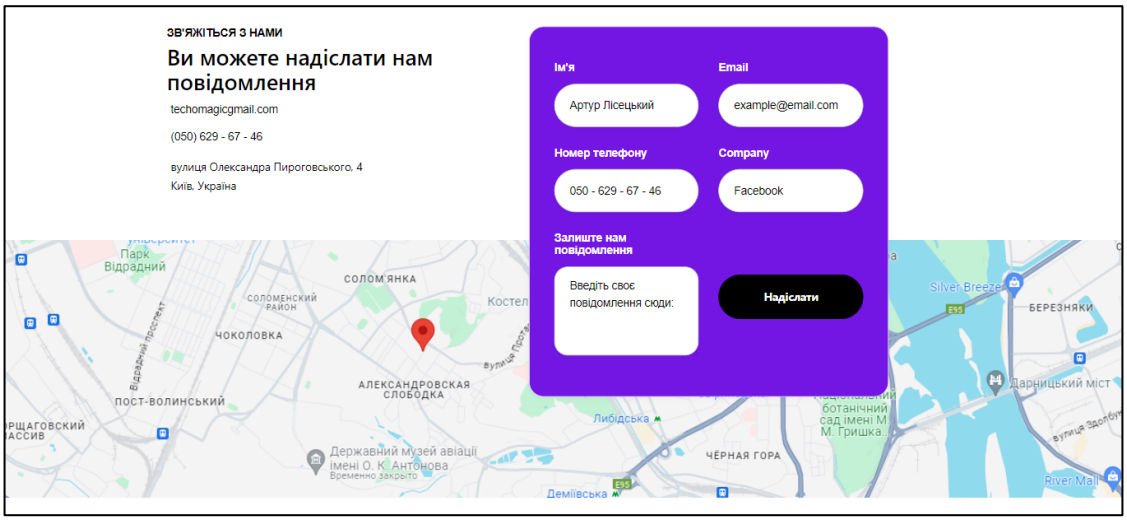


Рис. 3.9 Макет сторінки «Контакти»

На сторінці «Каталог» було розміщено перелік категорій товарів, які представлені у web-застосунку. Натискаючи на певну категорію товарів, користувач потрапляє до переліку товарів цієї категорії.

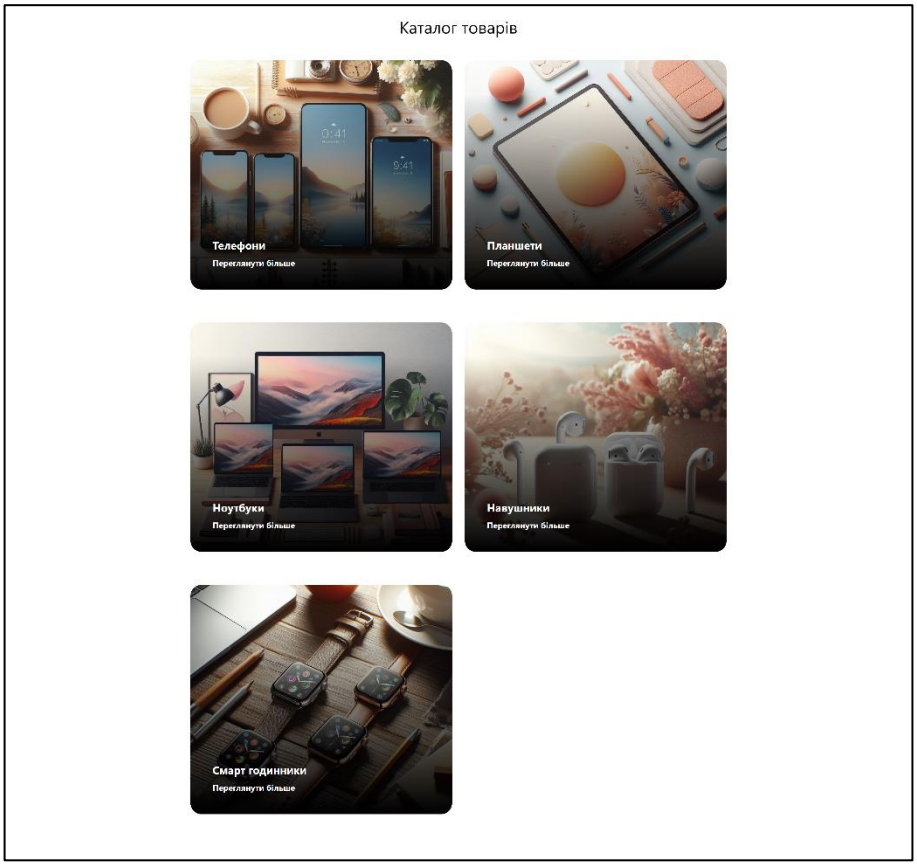


Рис. 3.10 Макет сторінки «Каталог»

На сторінці «FAQs» було розміщено перелік популярних запитань та відповідей на них, які дозволяють користувачу отримати необхідну інформацію у разі виникнення запитань.

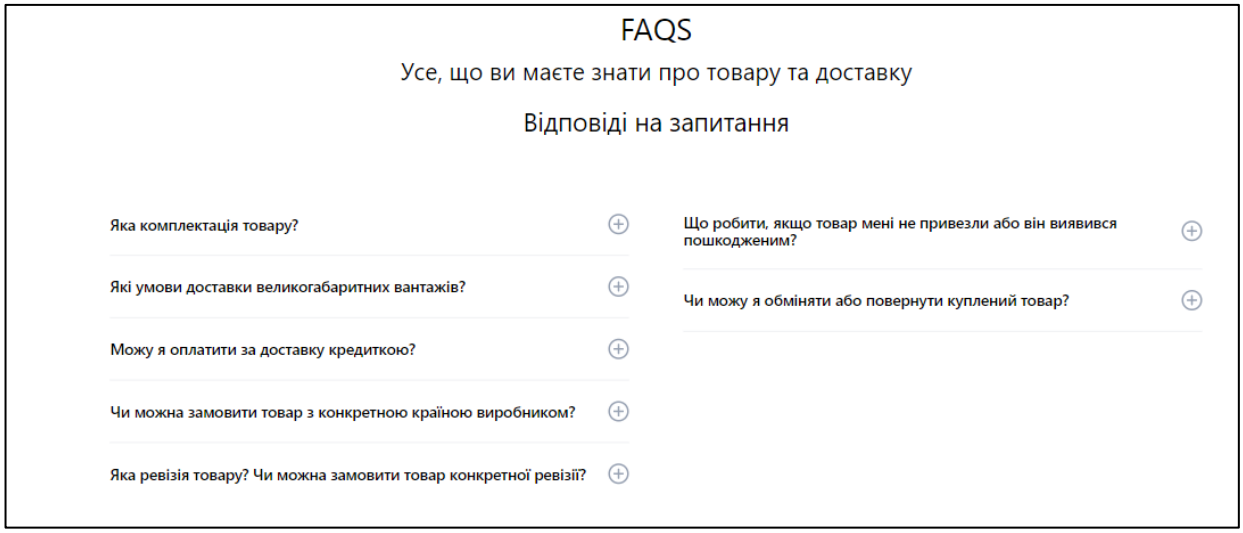


Рис. 3.11 Макет сторінки «FAQs»

На сторінці «Доставка і оплата» були розміщені текстові елементи, які містять інформацію про умови та терміни доставки і оплати замовлень.

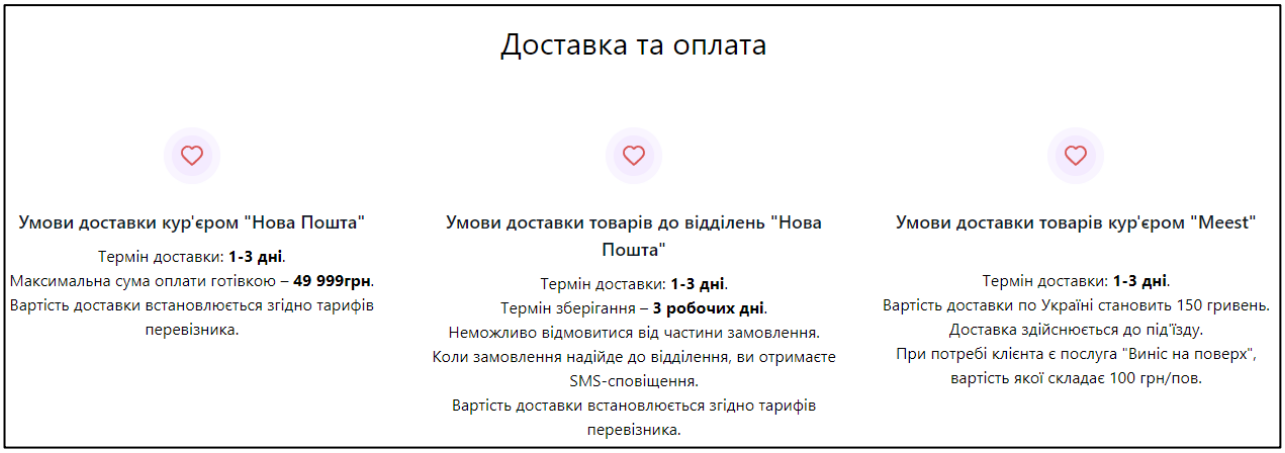


Рис. 3.12 Макет сторінки «Доставка і оплата»

На сторінці «Умови повернення» були розміщені текстові елементи, які містять актуальну інформацію про умови повернення товарів у разі браку.

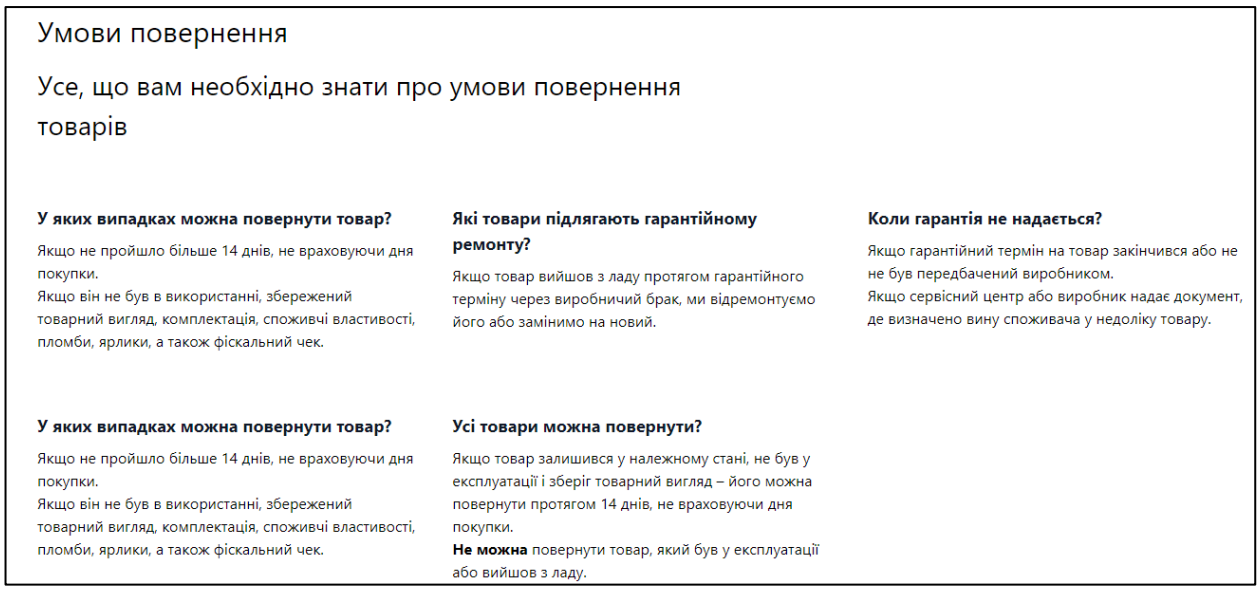


Рис. 3.13 Макет сторінки «Умови повернення»

На сторінці «Логін» були розміщені текстові елементи та форми для введення даних, у які користувач вводить дані для авторизації у web-застосунку. За допомогою кнопки «Логін» користувач проводить процедури авторизації та має змогу перейти до особистого кабінету користувача.

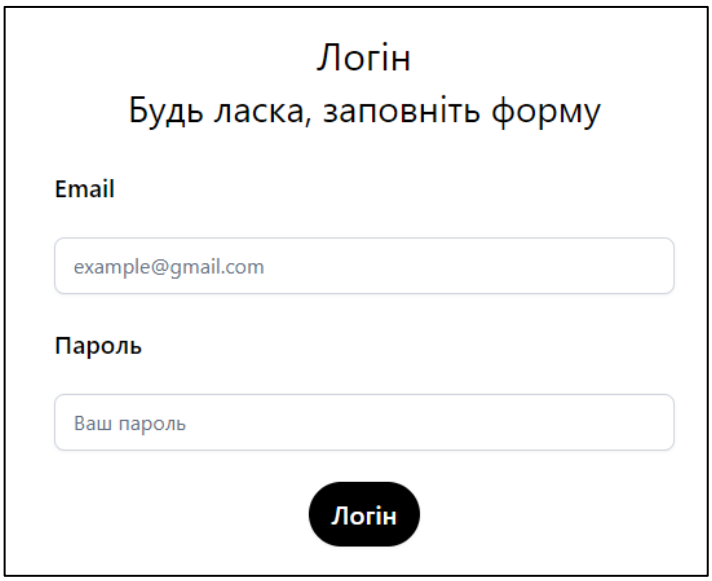


Рис. 3.14 Макет сторінки «Логін»

На сторінці «Реєстрація» були розміщені текстові елементи та форми для введення даних, у які користувач вводить дані для реєстрації у web-застосунку. За допомогою кнопки «Реєстрація» користувач проводить процедури реєстрації та має змогу перейти логіну.

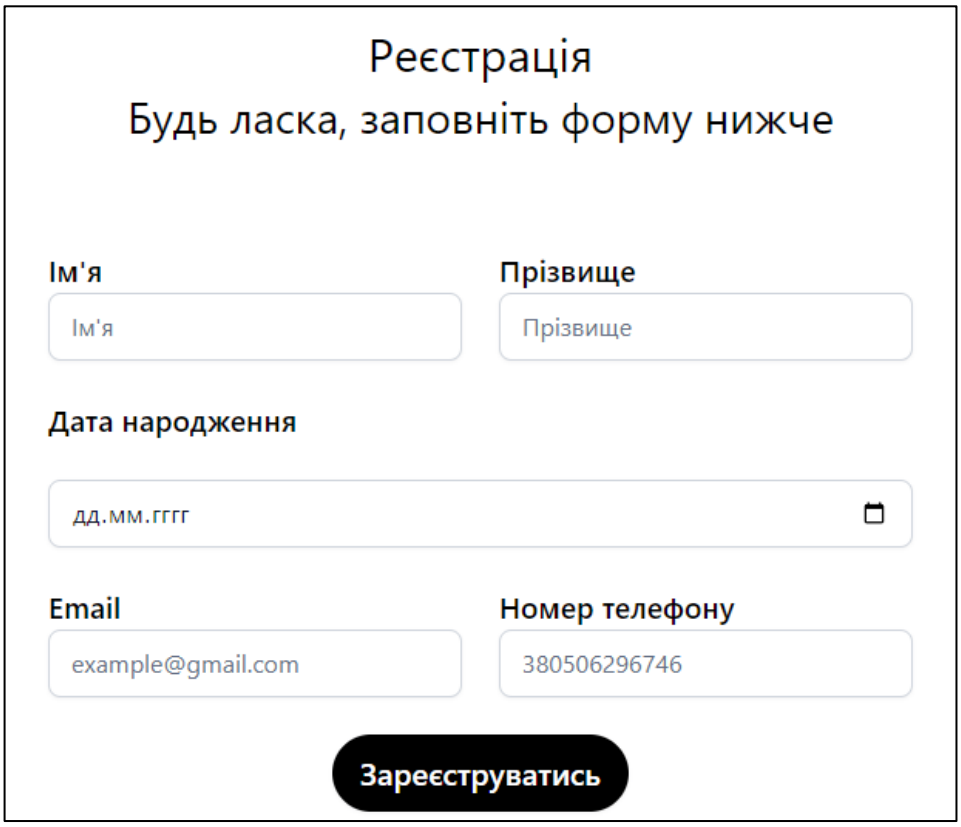


Рис. 3.15 Макет сторінки «Реєстрація»

На сторінці «Товари» було розміщено перелік товарів, які представлені у web-застосунку. З лівої сторони сторінки було розміщені елементи для керування фільтрами, які дозволяють провести фільтрацію необхідних товарів за заданими параметрами.

Біля кожного товару була розміщена кнопку «Купити», яка переміщує користувача на сторінку товару, для того, щоб користувач мав можливість отримати усю необхідну інформацію про товар. Кнопка «В кошик» додає товар у кошик користувача.

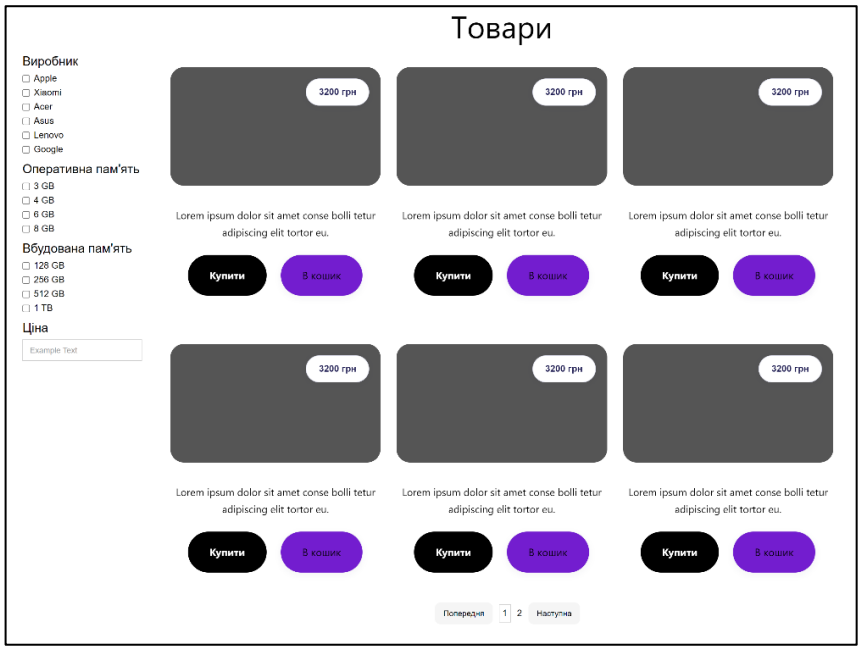


Рис. 3.16 Макет сторінки «Товари»

На сторінці товару було розміщено текстові елементи, які містять опис товару, форму для введення необхідної кількості товару, кнопки «Додати в кошик» та «Замовити». Кнопка «Додати в кошик» виконує процес додавання обраного товару у кошик користувача. Натискаючи кнопку «Замовити» користувач потрапляє до сторінки оформлення замовлення.

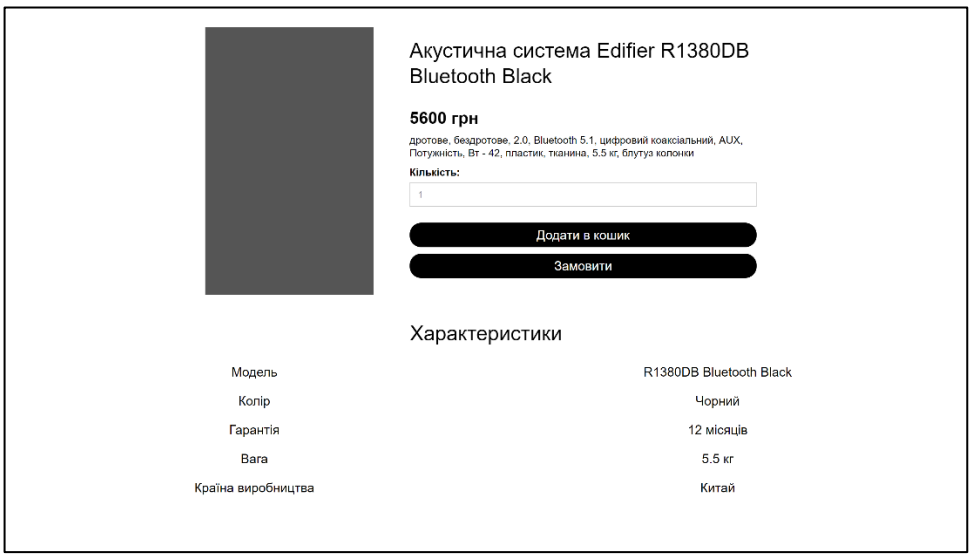


Рис. 3.17 Макет сторінки товару

На сторінці «Оформлення замовлення» було розміщено текстові елементи, які містять інформацію про товар та форми введення, у які користувач вводить дані для оформлення замовлення. Користувач має можливість обрати бажаний спосіб оплати. Присутня кнопка «Оформити замовлення», яка завершує процес оформлення замовлення.

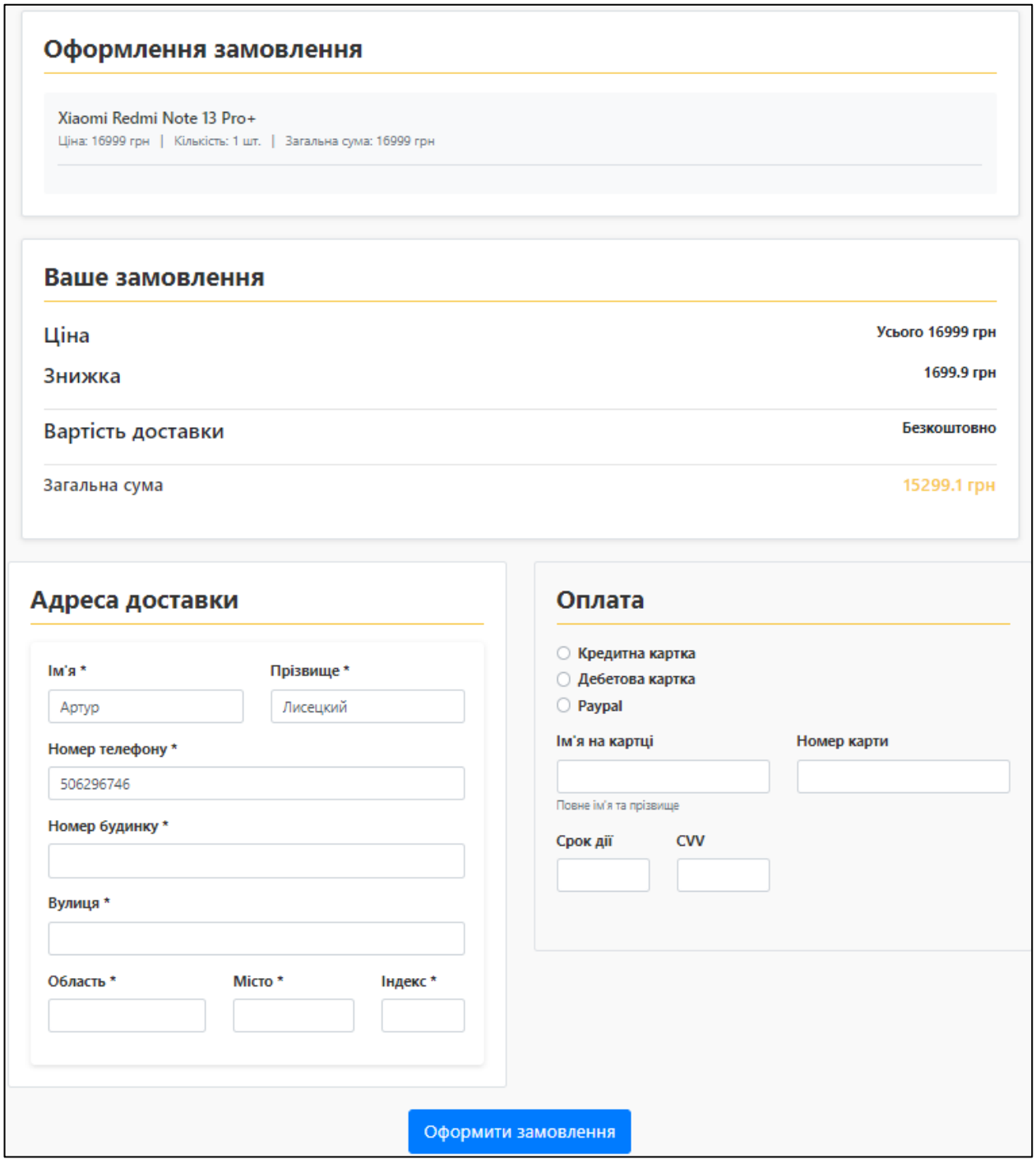


Рис. 3.18 Макет сторінки «Оформлення замовлення»

3.3 Реалізація клієнтської частини проекту

Для розробки клієнтської частини web-застосунку використовувались такі технології, як: HTML, CSS, JavaScript, TypeScript, Angular. За допомогою технологій HTML та CSS було створено розмітку для web-сторінок. Усі елементи були стилізовані та розміщені на сторінках у відповідних місцях.

За допомогою CSS [3] було реалізовано адаптивність web-застосунку, яка забезпечує коректне відображення усіх елементів на сторінках не залежно від того, який пристрій використовує користувач, наприклад: телефон, планшет, ноутбук або комп’ютер.

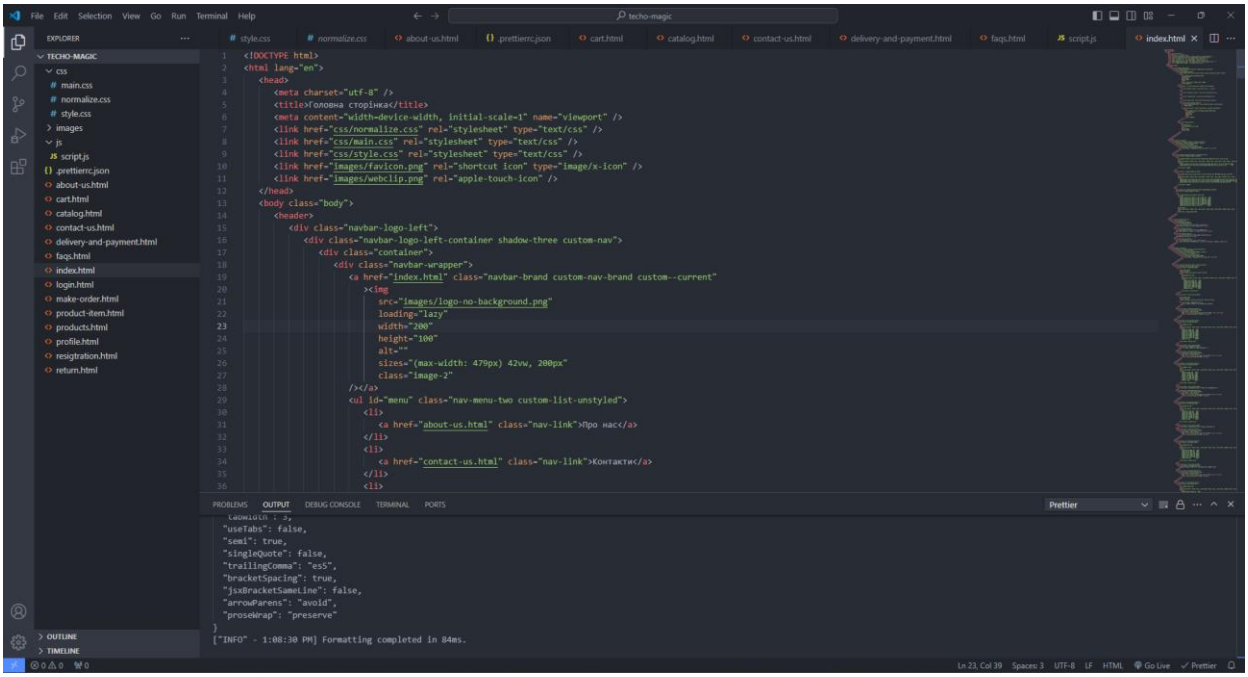
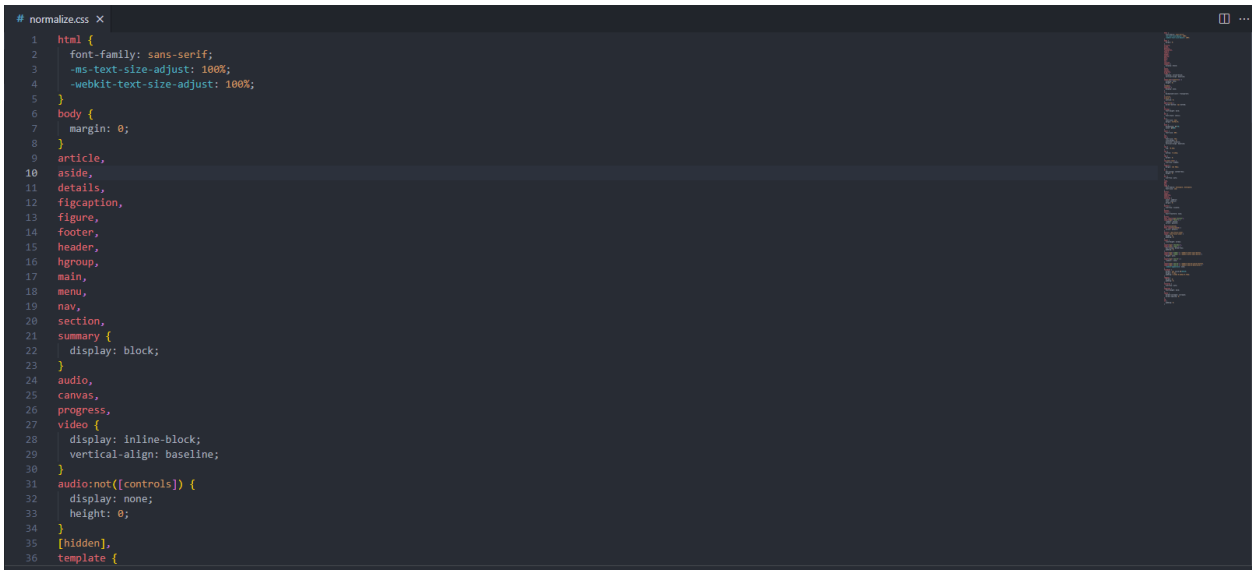


Рис. 3.19 Реалізація головної сторінки

Під час розробки були використані 3 CSS файли, які відповідають за графічне оформлення елементів на сторінках web-застосунку.

- Normalize.css містить початкові налаштування усіх елементів, прибираючи стандартні налаштування, які не будуть використовуватись.



```
# normalize.css ×
1  html {
2    font-family: sans-serif;
3    -ms-text-size-adjust: 100%;
4    -webkit-text-size-adjust: 100%;
5  }
6  body {
7    margin: 0;
8  }
9  article,
10 aside,
11 details,
12 figcaption,
13 figure,
14 footer,
15 header,
16 hgroup,
17 main,
18 menu,
19 nav,
20 section,
21 summary {
22   display: block;
23 }
24 audio,
25 canvas,
26 progress,
27 video {
28   display: inline-block;
29   vertical-align: baseline;
30 }
31 audio:not([controls]) {
32   display: none;
33   height: 0;
34 }
35 [hidden],
36 template {
```

Рис. 3.20 Код файлу Normalize.css

- Main.css містить загальні налаштування для елементів, таких як: body, img, h1, h2, h3, p.

```
# main.css 9+ ×
1  * {
2    -webkit-box-sizing: border-box;
3    -moz-box-sizing: border-box;
4    box-sizing: border-box;
5  }
6  html {
7    height: 100%;
8  }
9  body {
10   margin: 0;
11   min-height: 100%;
12   background-color: #fff;
13   font-family: Arial, sans-serif;
14   font-size: 14px;
15   line-height: 20px;
16   color: #333;
17 }
18 img {
19   max-width: 100%;
20   vertical-align: middle;
21   display: inline-block;
22 }
23 html.custom-mod-touch * {
24   background-attachment: scroll !important;
25 }
26 .custom-block {
27   display: block;
28 }
29 .custom-inline-block {
30   max-width: 100%;
31   display: inline-block;
32 }
33 .custom-clearfix:before,
34 .custom-clearfix:after {
35   content: " ";
36   display: table;
```

Рис. 3.21 Код файлу Main.css

- Style.css містить основні налаштування для елементів, які відображаються на сторінках web-застосунку.

Рис. 3.22 Код файлу Style.css

Для забезпечення швидкої та надійної роботи web-застосунку було використано фреймворк Angular до мови програмування JavaScript. Angular дозволяє створювати Single Page Applications, які можуть працювати без потреби оновлення сторінки. Усі елементи, необхідні для роботи web-застосунку зберігаються у компонентах, які містять окремі частини коду, який відповідає за відображення елементів на сторінках web-застосунку.

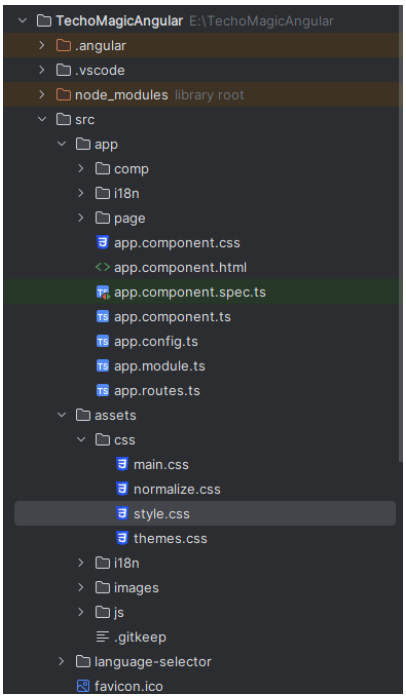


Рис. 3.23 Структура проекту Angular

Завдяки зручній організації файлів, значно покращується читабельність коду, оскільки усі сторінки складаються з окремих компонентів, які відповідають за окремі частини графічного інтерфейсу.

Header і Footer web-сторінок зберігається в окремих компонентах та використовуються лише один раз. Таким чином, зменшується кількість необхідного коду для роботи web-застосунку. Ці елементи завжди залишаються статичними та не змінюються при зміні сторінок web-застосунку.

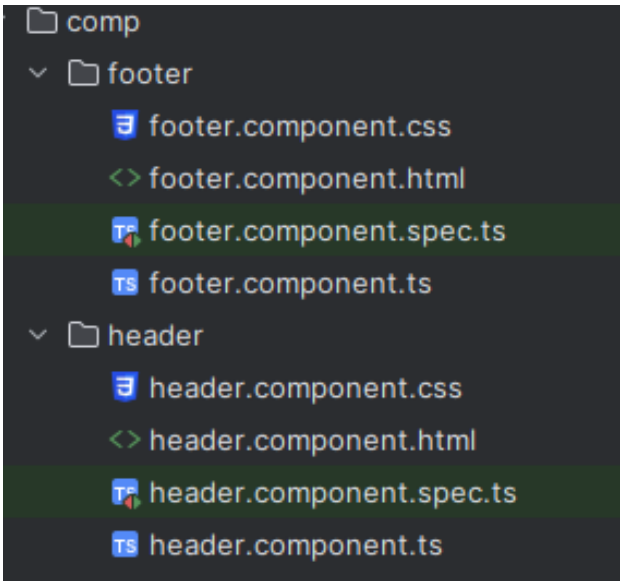


Рис. 3.24 Структура компонентів Header та Footer

Кожен компонент web-застосунку має свої файли для налаштування візуального відображення елементів, за які відповідає компонент та файли, у які відповідають за реалізацію функціональних можливостей елементів. Це дозволяє легко орієнтуватись у проекті під час розробки web-застосунку, оскільки усі необхідні налаштування знаходяться у відповідних компонентах, а не в одному файлі.

Для підключення усіх необхідних компонентів використовується файл App.component.html, який поєдную усі компоненти між собою.

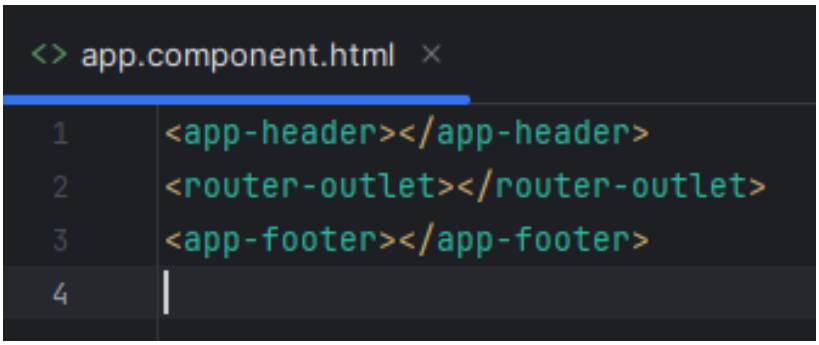


Рис. 3.25 Код файлу App.component.html

Пересування між сторінками web-застосунку реалізоване за допомогою елементів маршрутизації RouterLink, який відповідає за визначення адреси посилань.

Для коректної роботи була налаштована маршрутизація сторінками web-застосунку у файлі App.Routes.ts, який містить конфігурації маршрутів та визначає, які компоненти web-застосунку будуть відображатись при переході за URL-адресами.

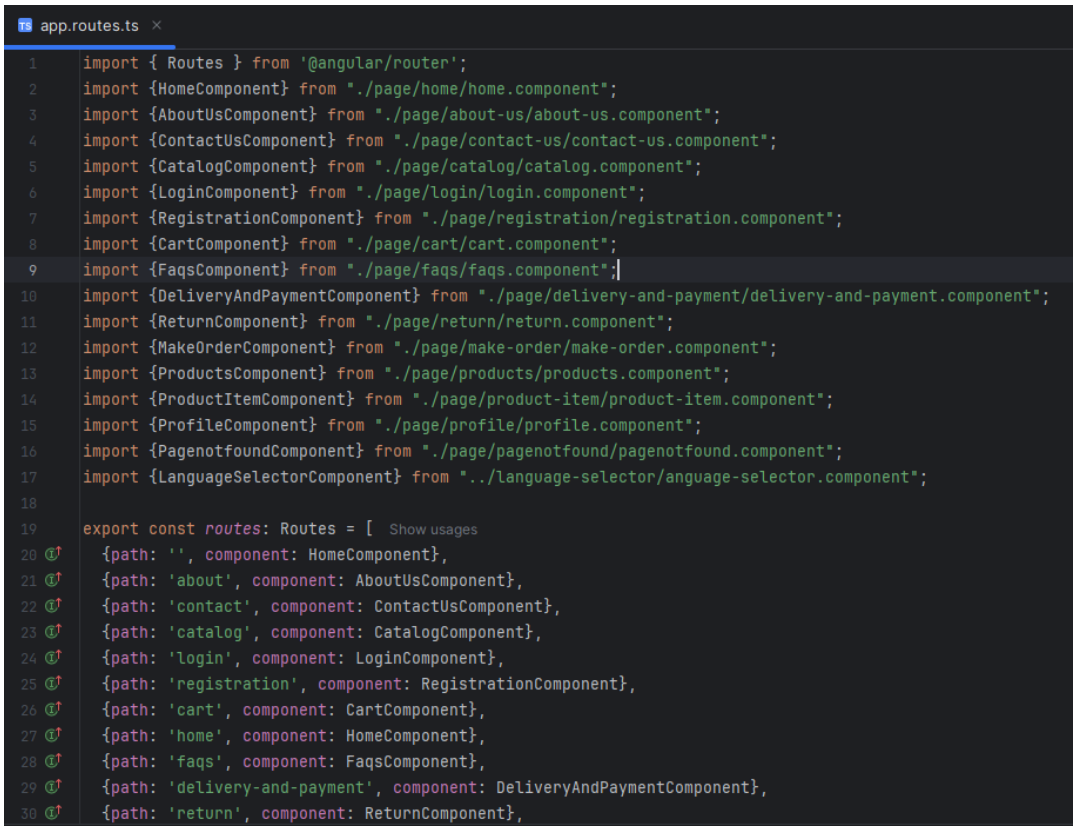


Рис. 3.26 Код файлу App.Routes.ts

Натискаючи на посилання RouterLink, користувач відправляє запит до файлу App.Router.ts, який визначає, які саме компоненти необхідно відобразити на сторінці web-застосунку.

```
<li class="menu-home">
  <a routerLink="home" class="nav-link">{{ 'main-page' | translate }}</a>
</li>
```

Рис. 3.27 Приклад використання RouterLink

Були реалізовані можливості зміни мови інтерфейсу користувача та зміни теми з світлої на темну. Зміна мови інтерфейсу дозволить іноземним користувачам вільно використовувати web-застосунок без необхідності використання стороннього перекладача.

Можливість змінити теми інтерфейсу користувача дає можливість обрати комфортну для користувача тему, що підвищить зручність використання web-застосунку.

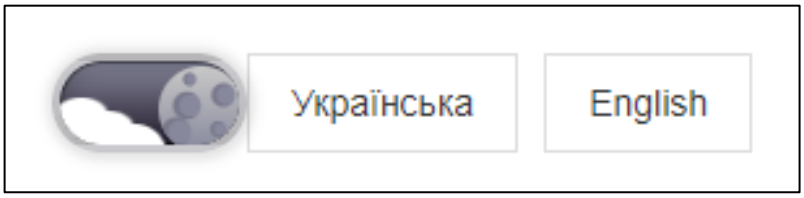


Рис. 3.28 Перемикачі теми та мови інтерфейсу

Перемикання теми реалізоване за допомогою перемикача, що визначає, яку кольорову палітру слід використовувати для відображення елементів на сторінці. Інформація про обрану тему зберігається у Cookie файли браузеру, що дозволяє запам’ятовувати обрані налаштування навіть при оновленні сторінки.

```
onThemeSwitchChange() : void  {  Show usages
  this.isLightTheme = !this.isLightTheme;
  this.cookieService.set( name: 'theme',  value: this.isLightTheme ? 'light' : 'dark');
  this.applyTheme();
}

private applyTheme() : void  {  Show usages
  document.body.setAttribute(
    qualifiedName: 'data-theme',
    value: this.isLightTheme ? 'light' : 'dark'
  );
}
```

Рис. 3.29 Реалізація перемикання теми інтерфейсу

Можливість зміни мови інтерфейсу користувача реалізована за допомогою перемикача, який визначає, яка мова інтерфейсу повинна бути використана на сторінці web-застосунку. Обираючи потрібну мову інтерфейсу, користувач надсилає запит до файлу language-selector.component.ts, що визначає, який саме набір текстових елементів повинен бути відображений.

```
import { Component } from '@angular/core';
import { LanguageService } from '../app/i18n/language.service';

@Component({  Show usages
  selector: 'app-language-selector',
  templateUrl: './language-selector.component.html'
})
export class LanguageSelectorComponent {
  languages : {code: string, label: string}[...  = [
    { code: 'ua', label: 'Українська' },
    { code: 'en', label: 'English' }
  ];

  constructor(private languageService: LanguageService) {}  no us

  switchLanguage(lang: string) : void  {  Show usages
    this.languageService.changeLanguage(lang);
  }
}
```

Рис. 3.30 Код файлу language-selector.component.ts

Набір текстових елементів, які відображаються на сторінках web-застосунку зберігається у json файлах, що дозволяють швидко отримати доступ до необхідних елементів для відображення на сторінці.

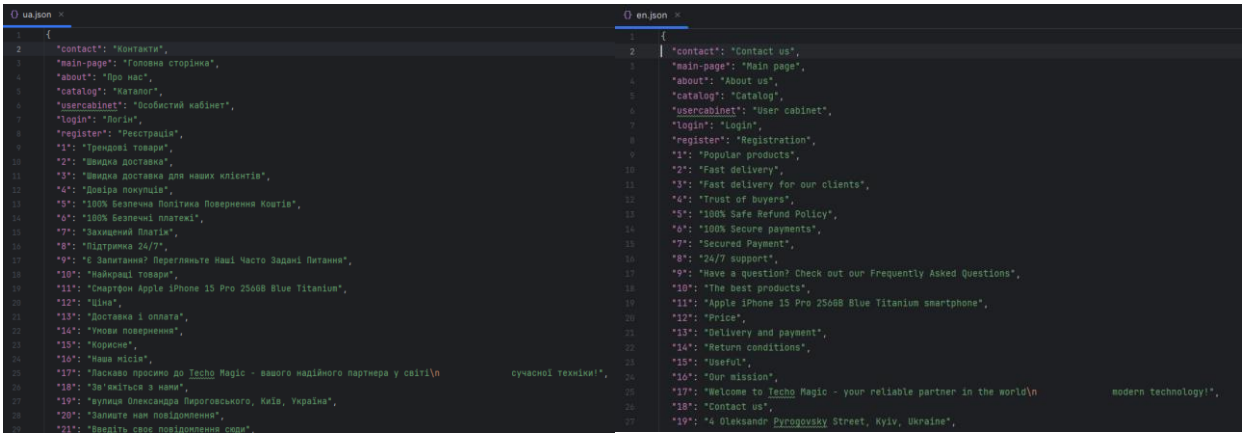


Рис. 3.31 Вміст файлу ua.json та en.json

Перевагами використання json файлів для зберігання наборів текстових даних є:

1. Зручність у використання, оскільки json файли мають простий та зрозумілий інтерфейс.
2. Легкість парсингу, оскільки json файли підтримуються багатьма мовами програмування.
3. Json файли мають маленький розмір, порівняно з іншими форматами, що робить процес передачі даних швидким.

Сторінка особистого кабінету користувача містить інформацію про користувача, список бажань, у який користувач може додати бажані товари та список минулих замовлень, який містить інформацію про товари, які були у замовленні.

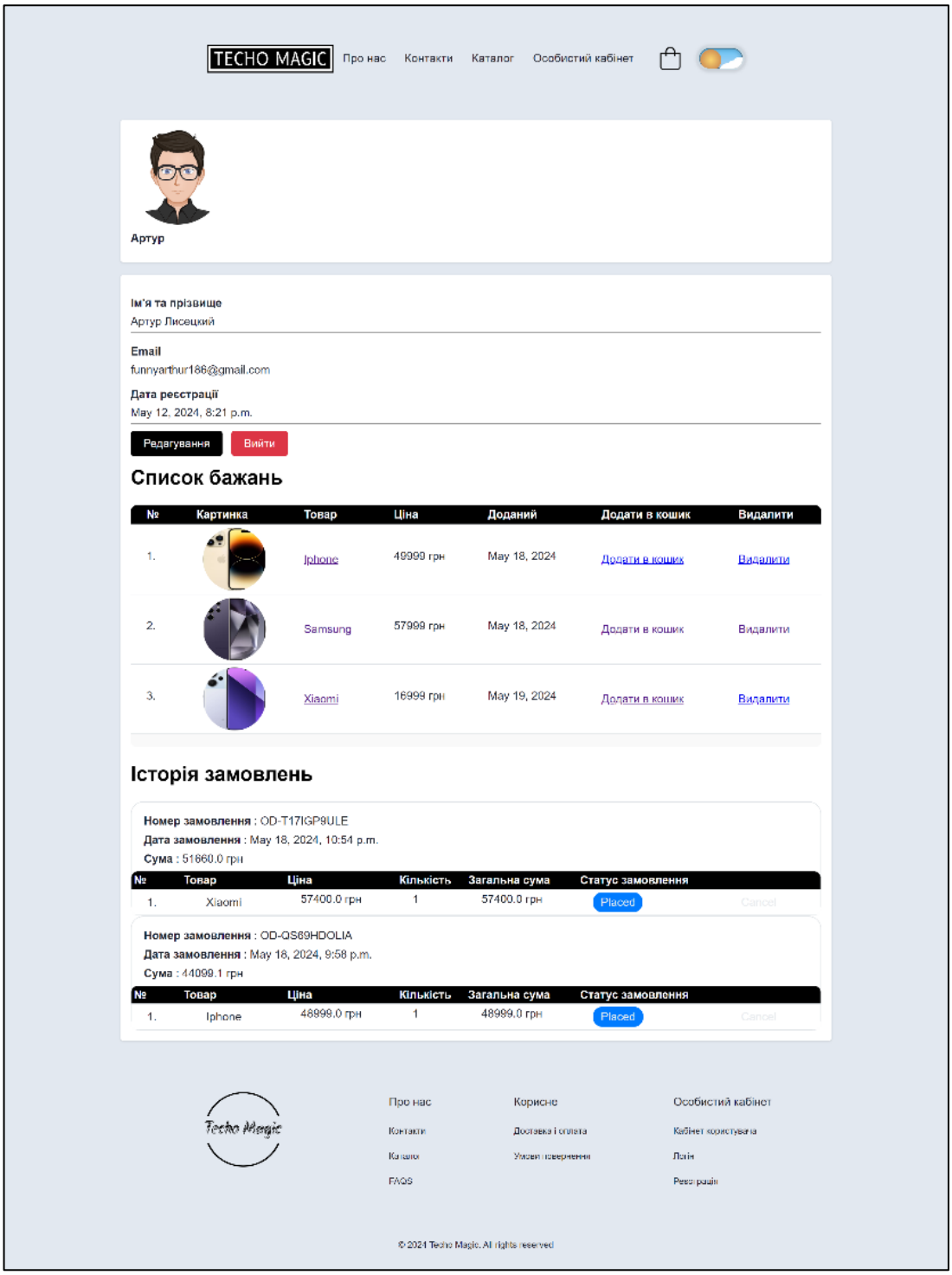


Рис. 3.32 Сторінка особистого кабінету користувача

На сторінці товарів, користувач може переглянути усі доступні товари, виконати фільтрацію товар за заданими параметрами, перейти на сторінку з детальною інформацією про товар. Також присутні кнопки для додавання товару у кошик та оформлення замовлення.

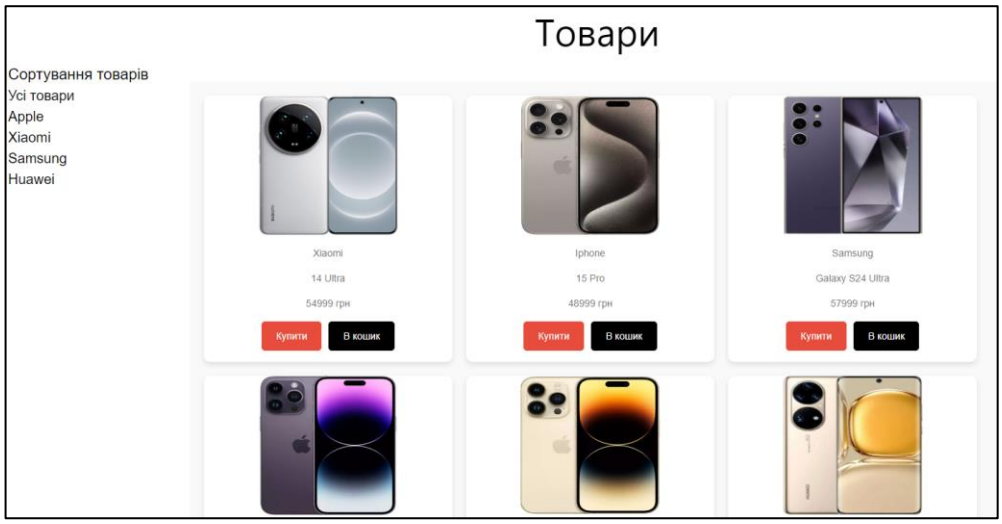


Рис. 3.33 Сторінка товарів

Сторінка з детальною інформацією про товар дає змогу користувачу прочитати детальну інформацію про товар, його характеристики. Присутні кнопки для додавання товару у список бажань, додавання товару у кошик та переходу до оформлення замовлення.

На сторінці присутнє фото товару, що дає можливість користувачу переглянути, як саме виглядає товар.

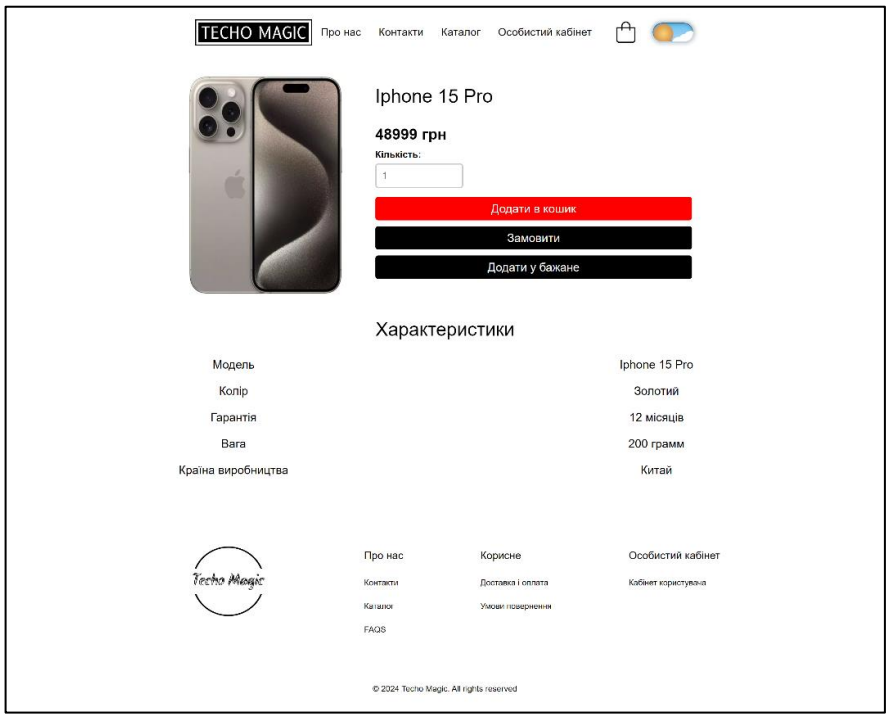


Рис. 3.34 Сторінка з детальною інформацією про товар

На сторінці кошику користувач може побачити інформацію про товар, який він додав у кошик, його ціну, кількість доданого товару. Присутня кнопка для видалення товару з кошику. Також на сторінці є кнопка для переходу до оформлення замовлення.

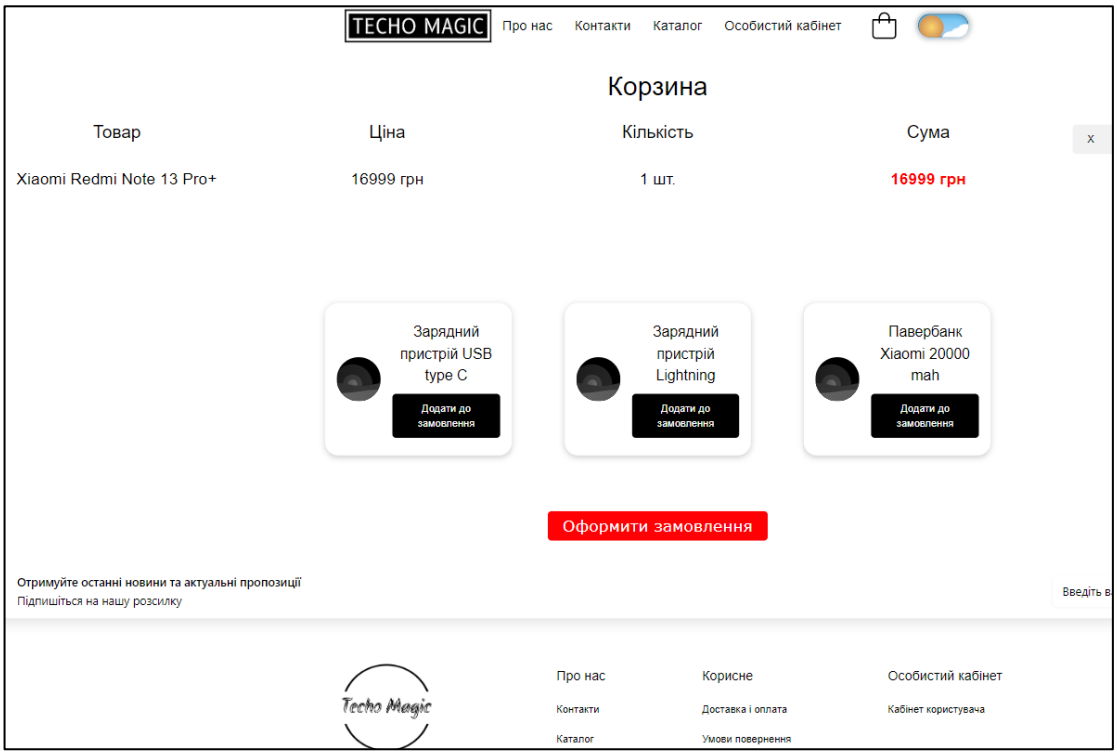


Рис. 3.35 Сторінка кошику

На сторінці оформлення замовлення присутні інформація про товар, який користувач додав у кошик, ціна товару, кількість товару та загальна ціна з урахуванням знижки. Присутні форми для введення особистої інформації про замовника та форми для вибори способу оплати замовлення.

Після того, як користувач ввід усі дані, відбувається процес валідації даних, який перевіряю їх коректність. Якщо усі дані введено вірно, замовлення оформлюється та користувач може побачити його у особистому кабінеті.

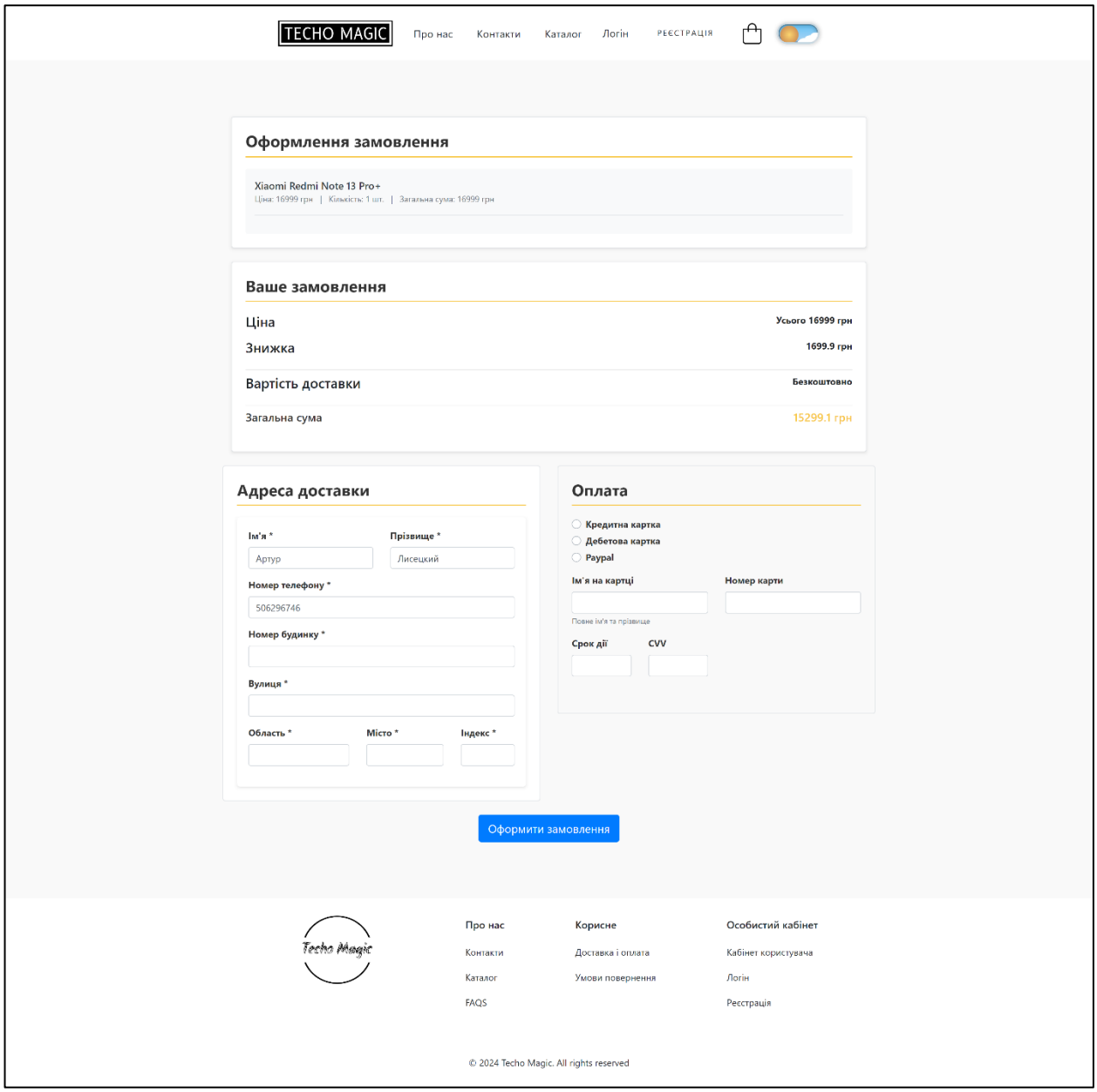


Рис. 3.36 Сторінка оформлення замовлення

3.4 Реалізація серверної частини проекту

Для реалізації серверної частини web-застосунку було обрано мову програмування Python та фреймворк Django, які дозволяють створити надійний web-застосунок, який має достатній рівень захищеності від різних типів атак та вірусів, що дозволяє забезпечити необхідний рівень безпеки конфіденційних даних користувачів.

Для зберігання даних було обрано систему управління базами даних SQLite, яка дозволяє додавати, редагувати та видаляти файли з бази даних та забезпечує необхідний рівень безпеки даних.

Схема бази даних містить інформацію про таблиці, які існують у базі даних. На рисунку 3.37 наведено схему розробленої бази даних, яка дозволяє зберігати, редагувати та видаляти інформацію з бази даних.

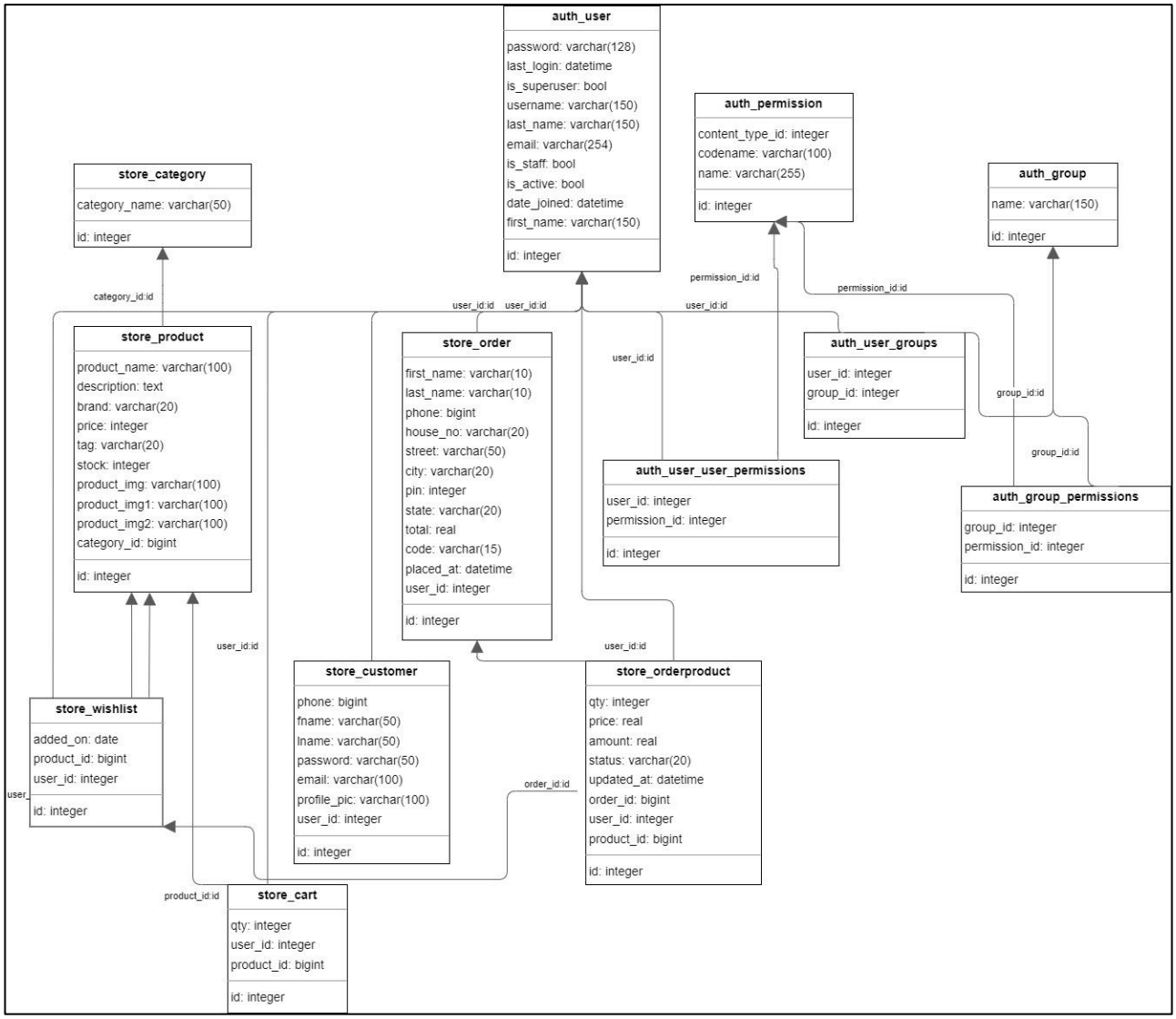


Рис. 3.37 Схема бази даних

Основними таблицями у базі даних є:

1. Customer – таблиця, яка відповідає за збереження даних користувача, його ім’я, прізвище, пароль, електронну пошту та унікальний id користувача, який він отримує при реєстрації.
2. Product – таблиця, що містить інформацію про товари, які представлені у web-застосунку, назву товару, опис, ціну, кількість товарів у наявності, картинки для відображення товару, категорії товарів та виробника.
3. Category – таблиця, у якій міститься інформацію про категорії товарів, які представлені у web-застосунку.
4. Cart – таблиця, яка зберігає дані про товар, коли користувач додає його у кошик.
5. Wishlist – таблиця, що відповідає за збереження даних про товари, які користувач додав у список бажань.
6. Order – таблиця, яка відповідає за збереження даних користувача, які вводяться при оформлені замовлення. Таблиця містить поля для збереження інформацію про ім’я та прізвище замовника замовлення, номер телефону, назву вулиці, номер будинку, місто, індекс, область, та час, коли було оформлено замовлення.
7. Auth.user – таблиця, які містить інформацію про користувача, його пароль, ім’я, прізвище, електронну пошту та дату реєстрації.

Можливість реєстрації особистого кабінету користувача реалізована за допомогою вбудований у Django модулів, які відповідають за процеси реєстрації та авторизації користувача у особистий кабінет.

Користувачу для реєстрації необхідно ввести свої ім’я, прізвище, номер телефону, електронну пошту та дату народження.

Після введення користувач натискає кнопку «Реєстрація», введенні дані перевіряються та надсилається запит у базу даних для додавання інформації у таблиці.

Після завершення процесу реєстрації користувач має можливість одразу перейти до особистого кабінету.

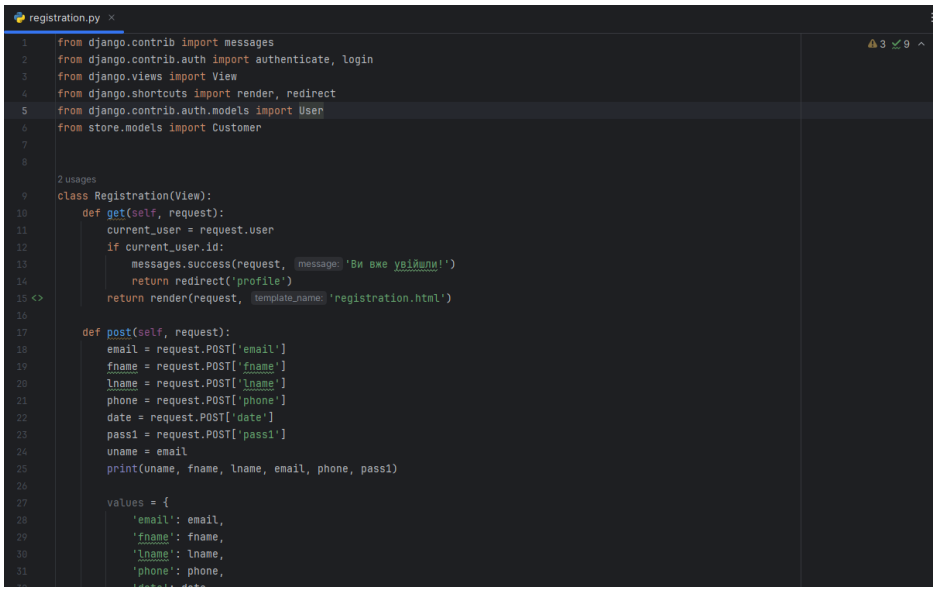


Рис. 3.38 Реалізація реєстрації користувача

Авторизація зареєстрованого користувача реалізована за допомогою вбудований у Django модулів авторизації, які дозволяють перевірити введені користувачем дані у поля авторизацію. Якщо користувач ввів валідні дані, які відповідають даним, які були введенні при реєстрацію, користувач авторизується та може перейти до особистого кабінету. Якщо дані були введенні не вірно, система сповіщає користувача про те, що логін або пароль не вірні.

Для забезпечення необхідно рівня безпеки, авторизація користувача проводиться за допомогою CSRF-токену, який створюється за надсилається у браузер користувача. Якщо токен, який отримав браузер і токен користувача однакові, система продовжую працювати.

CSRF-токен – це випадкове секретне значення, яке забезпечує необхідний рівень безпеки при роботі з web-застосунок. Він є обов’язковим для усіх форм, у які користувач може ввести свої дані. Без цього токену не буде працювати не тільки система авторизації, а й інші системи розробленого web-застосунку.

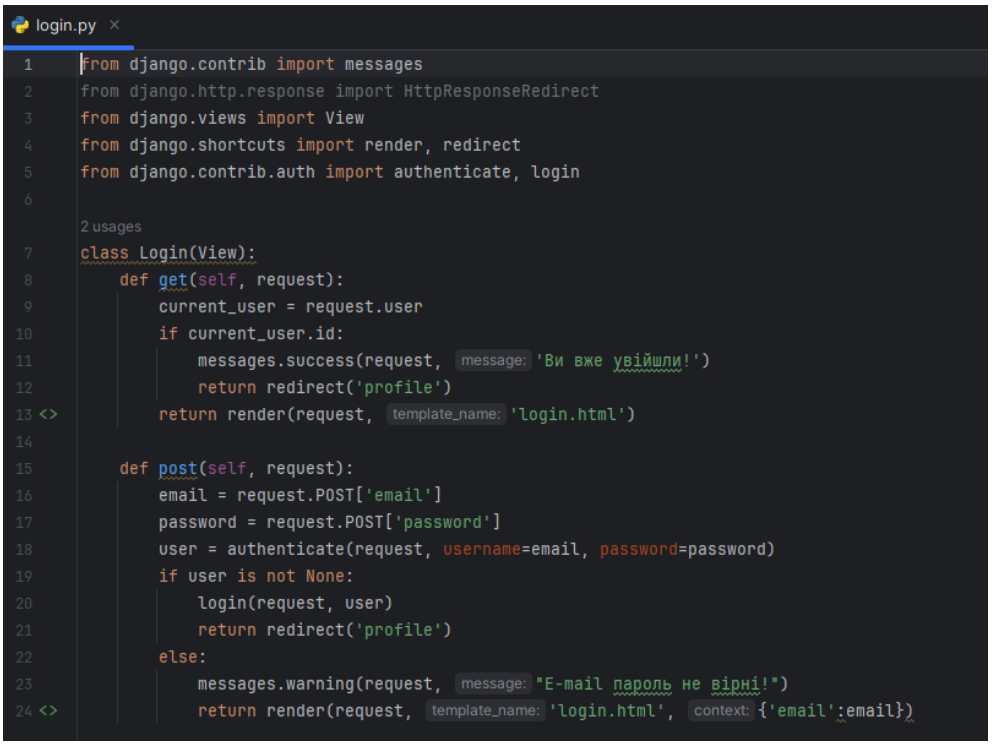


Рис. 3.39 Реалізація авторизації користувача

У особистому кабінеті користувач може переглянути особисту інформацію, дані про товари, які були додані у список бажань та список минулих замовлень, які робив користувач. Також присутня можливість вийти з особистого акаунту та змінити інформацію про користувача, яка була введена при реєстрації.

Товари, які відображаються у списку бажань можна додати до кошику або видалити зі списку бажань. Натиснувши на назву товару, який доданий у список бажань, користувач має можливість перейти на сторінку цього товару, щоб ознайомитись з його характеристиками.

Для реалізації особистого кабінету користувача було використано функцію Account, яка перевіряє, чи авторизований користувач та надсилає запит у базу даних та відповідає за відображення необхідної інформацію про користувачі, список бажань та минулі замовлення.

Якщо користувач не робив замовлення та не додавав товари у список бажань, відображається повідомлення про те, що у користувача ще не було замовлень та список бажань пустий.

```python
@login_required(login_url='/login')
def account(request):
    orderprs = []
    currentuser = request.user

    carts = Cart.objects.filter(user_id=currentuser.id)
    qty = 0
    total = 0
    for cart in carts:
        total = total + cart.amount
        qty = qty + cart.qty

    categories = Category.objects.all()
    customer = Customer.objects.get(user_id=currentuser.id)
    wishlists = Wishlist.objects.filter(user_id=currentuser.id)

    orders = Order.objects.filter(user_id=currentuser.id).order_by("-placed_at")
    for order in orders:
        pr = OrderProduct.objects.filter(order_id=order.id)
        orderprs.append(pr)

    details = {
        'customer': customer,
        'orders': orders,
        'orderprs': orderprs,
        'qty': qty,
        'total': total,
        'carts': carts,
        'categories': categories,
        'wishlists': wishlists
    }
```

Рис. 3.40 Реалізація особистого кабінету користувача

На сторінці товарів, користувач може переглянути усі товари, які представлені у web-застосунки, або відфільтрувати їх, обравши необідні параметри для фільтрації. У полі для задання фільтрів користувач може обрати того виробника, товари якого він бажає придбати і необхідні товари будуть відображені на сторінці web-застосунку.

Користувач може одразу перейти до замовлення товару, без переходу на сторінку з детальною інформацією про товар, або додати товар у кошик.

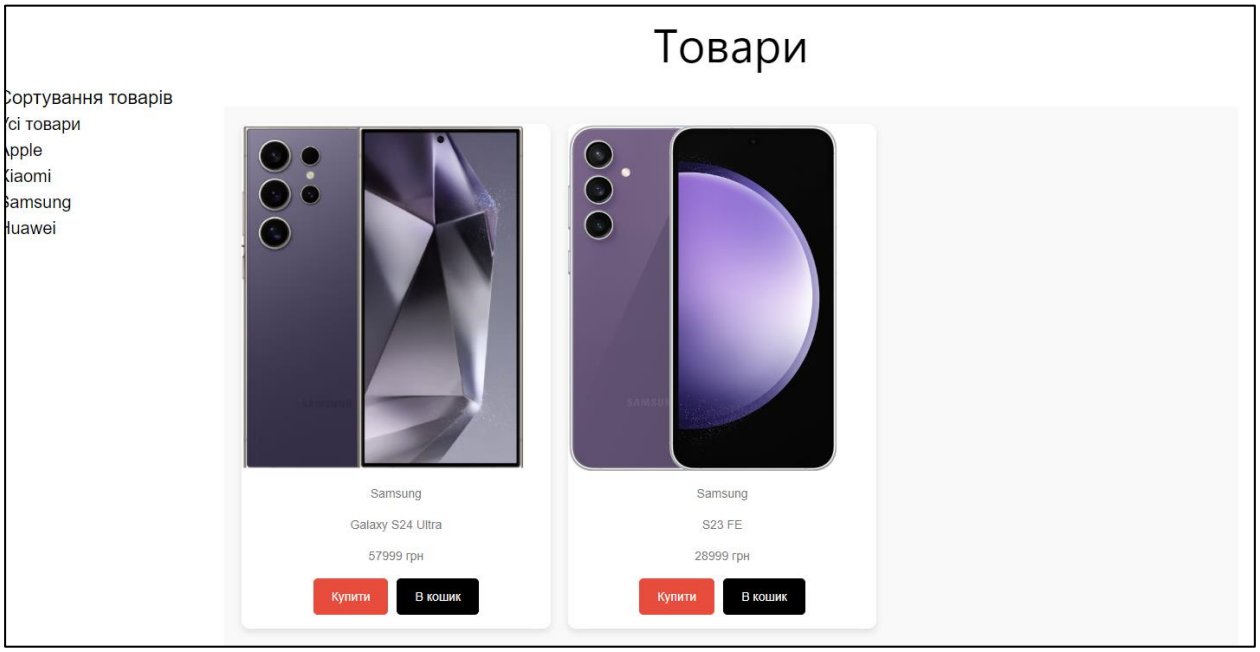


Рис. 3.41 Сторінка товарів після застосування фільтрів

Для реалізації можливість фільтрування товарів було використано категорії товарів, які відповідають запиту користувача. Користувач має можливість скинути налаштування фільтрів та показати усі товари, які представлені у web-застосунку.

```
def products(request):
    current_user = request.user
    customer = []
    try:
        customer = Customer.objects.get(user_id=current_user.id)
    except:
        pass
    categories = Category.objects.all()

    categoryid = 0
    filtered_category = []


    try:
        categoryid = request.GET['category']
    except:
        pass

    if categoryid:
        products = Product.objects.filter(category_id=categoryid)
        filtered_category = Category.objects.get(id=categoryid)
    else:
        products = Product.objects.all()
```

Рис. 3.42 Реалізація можливості застосування фільтрів

На сторінці з детальною інформацією про товари користувач може обрати необхідну кількість товару, додати товар у кошик, перейти до оформлення замовлення або додати товар у список бажань, який відображається у особистому кабінету користувача.

```python
def productdetail(request, prid):
    current_user = request.user
    product = Product.objects.get(id=prid)
    carts = Cart.objects.filter(user_id=current_user.id)
    wishlist = Wishlist.objects.filter(user_id=current_user.id, product_id=prid)

    customer = []
    try:
        customer = Customer.objects.get(user_id=current_user.id)
    except:
        pass

    try:
        pr_qty = Cart.objects.get(user_id=current_user.id, product_id=prid)
        pr_qty = pr_qty.qty
    except:
        pr_qty = 0

    qty = 0
    total = 0
    for cart in carts:
        total = total + cart.amount
        qty = qty + cart.qty

    details = {
        'customer': customer,
        'product': product,
        'qty': qty,
        'total': total,
        'carts': carts,
        'wishlist': wishlist,
        'pr_qty': pr_qty
```

Рис. 3.43 Реалізація сторінки з детальною інформацією про товар

Після додавання товару у кошик, надсилається запит у базу даних для додання інформації про товар у необхідні таблиці. Користувач може перейти до сторінки кошики, на якій буде відображена інформація про товар, що був доданий у кошик. Користувач може видалити товар з кошику, додати до кошики додаткові товари або перейти до оформлення замовлення.

Сторінка кошику реалізована за допомогою функції Cart, яка перевіряє чи є користувач авторизованим. Якщо користувач авторизований, функція обраховує загальну ціну замовлення, ціну з урахуванням знижки та відповідає за відображення елементів на сторінці кошику web-застосунку.

```python
@login_required(login_url='/login')
def cart(request):
    current_user = request.user
    customer = Customer.objects.get(user_id=current_user.id)
    carts = Cart.objects.filter(user_id=current_user.id)

    total = 0
    qty = 0
    for cart in carts:
        total = total + cart.amount
        qty = qty + cart.qty

    discount = 10 / 100 * total
    if discount > 20000:
        discount = 20000

    grand_total = total - discount

    cart = {
        'customer': customer,
        'carts': carts,
        'qty': qty,
        'total': total,
        'discount': discount,
        'grand_total': grand_total,
    }

    return render(request, template_name: 'cart.html', cart)
```

Рис. 3.44 Реалізація сторінки кошику

Якщо користувач не був авторизований, він буде направлений на сторінки авторизація, і тільки після того, як користувач авторизується від зможе оформити замовлення. Після того, як користувач натискає кнопку оформлення замовлення, він переходить на сторінку «Оформлення замовлення», на якій відображається інформація про товар, який був доданий у кошик та поля введення інформації про замовника товару, інформації про адресу доставки та способи оплати.

Користувач має можливість оплатити замовлення онлайн, використовуючи форму обрання та введення способу оплати. Після того, як користувач оплатив замовлення, інформацію про замовлення додається у базу даних та відображається у особистому кабінету користувача у списку минулих замовлень.

```python
@login_required(login_url='/login')
def checkout(request):
    currentuser = request.user
    carts = Cart.objects.filter(user_id=currentuser.id)
    customer = Customer.objects.get(user_id=currentuser.id)
    myuser = User.objects.get(id=currentuser.id)

    total = 0
    qty = 0
    for cart in carts:
        total = total + cart.amount
        qty = qty + cart.qty

    discount = 10 / 100 * total
    if discount > 20000:
        discount = 20000

    grand_total = total - discount

    if request.method == "POST":
        firstname = request.POST['firstname']
        lastname = request.POST['lastname']
        phone = request.POST['phone']
        house_no = request.POST['house_no']
        street = request.POST['street']
        state = request.POST['state']
        city = request.POST['city']
        pin = request.POST['pin']
        code = "OD-" + get_random_string(10).upper()

        order = Order(
            user_id=currentuser.id,
            first_name=firstname,
```

Рис. 3.45 Реалізація оформлення замовлення

4 ТЕСТУВАННЯ WEB-ЗАСТОСУНКУ

4.1 Реалізація тестування web-застосунку

Після розробки web-застосунку необхідно провести тестування, для того, щоб перевірити, як система працює для виявлення можливих помилок. Для тестування web-застосунку було Unit тестування, що забезпечує процес тестування кожного функціонального модулю web-застосунку окремо, дозволяючи перевірити як працюють модулі окремо один від одного.

Для тестування використовувались тести, реалізація яких представлена на малюнку 4.1

```
import { TestBed } from '@angular/core/testing';
import { RouterTestingModule } from '@angular/router/testing';
import { AppComponent } from './app.component';

describe( description: 'AppComponent', specDefinitions: () : void => {
  beforeEach( action: async () : Promise<void> => {
    await TestBed.configureTestingModule( moduleDef: {
      imports: [RouterTestingModule, AppComponent],
    }).compileComponents();
  });

  it( expectation: 'should create the app', assertion: () : void => {
    const fixture : ComponentFixture<AppComponent> = TestBed.createComponent(AppComponent);
    const app : AppComponent = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it( expectation: `should have the 'TechoMagicAngular' title`, assertion: () : void => {
    const fixture : ComponentFixture<AppComponent> = TestBed.createComponent(AppComponent);
    const app : AppComponent = fixture.componentInstance;
    expect(app.title).toEqual( expected: 'TechoMagicAngular');
  });

  it( expectation: 'should render title', assertion: () : void => {
    const fixture : ComponentFixture<AppComponent> = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled : HTMLElement = fixture.nativeElement as HTMLElement;
    expect(compiled.querySelector( selectors: 'h1')?.textContent).toContain( expected: 'Hello, TechoMagicAngular');
```

Рис. 4.1 Реалізація тестування web-застосунку

Після проведених тестувань було отримано наступні результати:

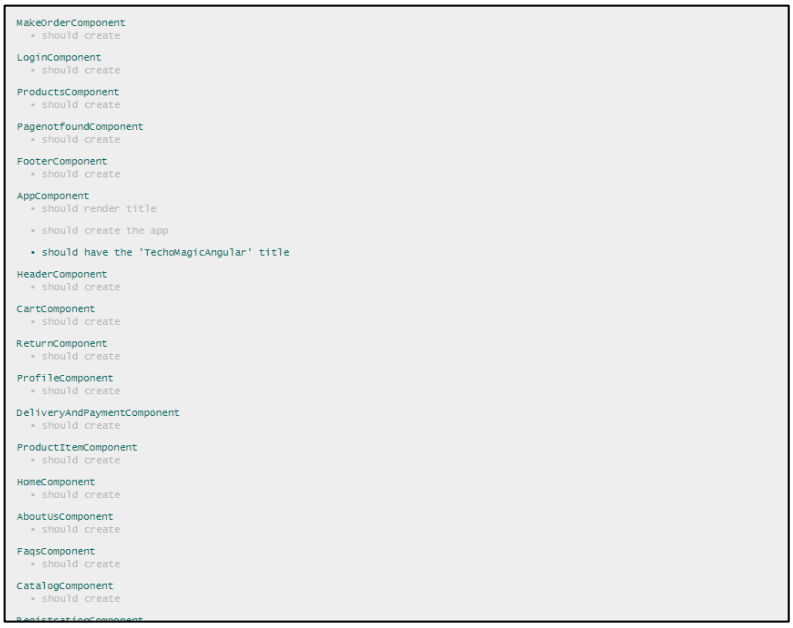


Рис. 4.2 Результати Unit-тестування

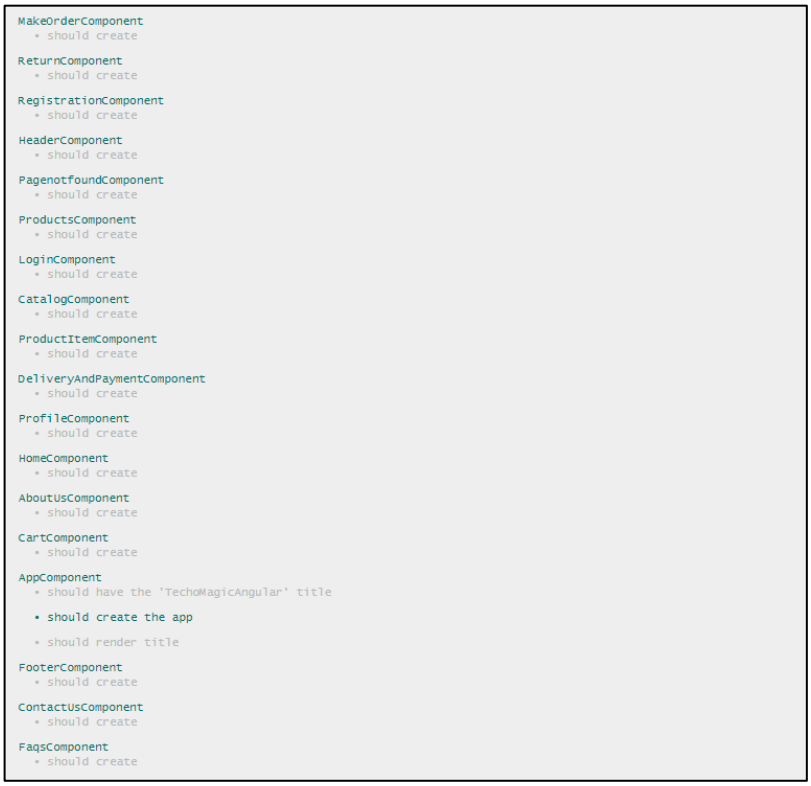


Рис. 4.3 Результати Unit-тестування

За результатами тестування помилок у роботі web-застосунку не було виявлено, що забезпечує користувачам надійність роботи web-застосунку.

ВИСНОВКИ

1. Під час аналізу задач були визначені функціональні та нефункціональні вимоги до роботи web-застосунку.
2. Аналіз існуючих аналогів показав сильні та слабкі місця web-застосунків з продажу техніки. Було визначено ряд особливостей, відсутніх у існуючих web-застосунків, такі як: можливість зміни мови інтерфейсу, можливість зміни теми інтерфейсу користувача, маленька кількість кроків для оформлення замовлення, що підкреслило унікальність розробки.
3. Проведено розробку архітектури системи, яка дозволила визначити, як повинна працювати система.
4. Проведено розробку клієнтської частини web-застосунку, яка включала створення макету інтерфейсу, реалізацію інтерфейсу користувача.
5. Проведено розробку серверної частини web-застосунку, яка включала створення бази даних, реалізацію функціональних вимог web-застосунку.
6. Проведене тестування web-застосунку за допомогою Unit-тестів підтвердило відсутність помилок при роботі web-застосунку та надійну працездатність системи.
7. Пройдено апробацію результатів дослідження:
    - Лісецький А.О., Замрій І.В. Визначення вимог до web-застосунку з продажу техніки із онлайн-системою оплати. Всеукраїніська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 248-249.
    - Лісецький А.О., Замрій І.В. Безпека web-застосунку з продажу техніки з онлайн-системою оплати. Всеукраїніська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, Державний університет

інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 253-54.

- Лісецький А.О., Замрій І.В. Можливості та переваги web-застосунку для продажу техніки з онлайн-системою оплати. Всеукраїніська науково-технічна конференція «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 24 квітня 2024 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С. 423-424.

ПЕРЕЛІК ПОСИЛАНЬ

1. Campesato O. Angular and Deep Learning Pocket Primer. Mercury Learning & Information, 2020. 250 p.
2. Fain Y., Moiseev A. TypeScript Quickly. Manning Publications Co. LLC, 2020.
3. Kaur G. Css: What Is CSS?. Independently Published, 2022.
4. Kumar R. Complete HTML Practical Course: HTML. Independently Published, 2020.
5. O'Neil T. Beginners Guide to Figma: The Step by Step Figma Manual with Illustrations and Tips. Independently Published, 2022.
6. Vickler A. Javascript: Javascript Basics for Beginners. Independently Published, 2021.
7. Vincent W. S. Django for Beginners: Build Websites with Python and Django. Welcometocode, 2020. 292 p.
8. Інтернет-магазин Foxtrot.com.ua. URL: https://www.foxtrot.com.ua/ (дата звернення 14.05.2024)
9. Інтернет-магазин Moyo.ua. URL: https://www.moyo.ua/ua/ (дата звернення 15.05.2024).
10. Інтернет-магазин Citrus.com.ua. URL: https://www.ctrs.com.ua/ (дата звернення 16.05.2024)

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка Web-застосунку з продажу техніки з онлайн-системою оплати за допомогою HTML, CSS, Python Django, Angular

Виконав студент 4 курсу
Групи ПД-42
Лісецький Артур Олександрович
Керівник роботи
Д.т.н., доцент, завідувач кафедри ІПЗ Замрій Ірина Вікторівна

Київ – 2024

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** - спрощення процесу продажу техніки онлайн за рахунок застосування веб-застосунку, створеного за допомогою HTML, CSS, Python Django, Angular.
- **Об'єкт дослідження -** процес продажу техніки онлайн за допомогою веб-застосунків.
- **Предмет дослідження -** автоматизація продажу техніки онлайн за допомогою веб-застосунків.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Провести аналіз існуючих web-застосунків з продажу техніки , визначити їх переваги та недоліки
2. Спроектувати архітектуру web-застосунку, забезпечити безпеку персональних даних користувача
3. Розробити візуальну частину web-застосунку та інтерфейс користувача
4. Розробити серверну частину web-застосунку
5. Провести тестування web-застосунку на відповідність вимогам

3

АНАЛІЗ АНАЛОГІВ

Характеристика	Foxtrot	Moyo	Citrus	TechoMagic
Мінімальна кількість кроків для замовлення товару	3	4	4	2
Підтримка англійської мови	-	-	-	+
Інтегрована система оплати	+	+	+	+
Адаптація інтерфейсу для різних пристроїв	+	+	+	+
Швидкість завантаження сторінок	3880 мс	6800 мс	4200 мс	2600 мс

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги

1. Можливість реєстрації та авторизації користувачів
2. Перегляд та редагування профілю користувача
3. Пошук необхідних товарів у каталозі товарів
4. Можливість додавання товарів у кошик та видалення їх
5. Можливість додавання товарів у список бажань
6. Можливість оформлення замовлення
7. Перегляд історії замовлень

Нефункціональні вимоги

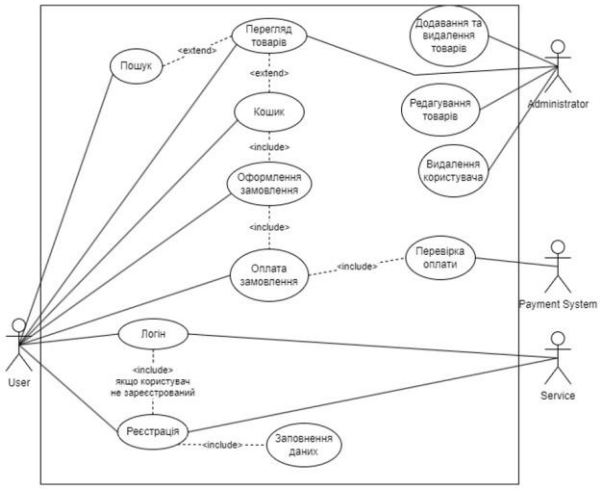
1. Забезпечення швидкодії web-застосунку
2. Використання елементів адаптивного верстання
3. Локалізація двома мовами (українська, англійська)
4. Можливість оплати замовлення онлайн
5. Мінімальна кількість кроків для оформлення замовлення: 2

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

6

Діаграма варіантів використання

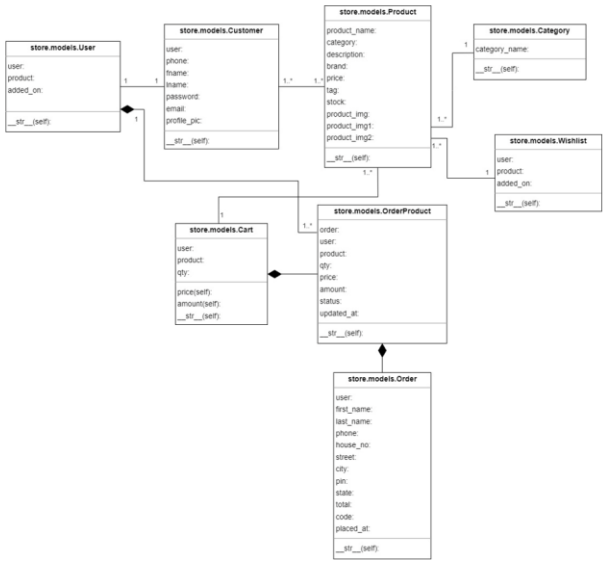


7

Схема роботи веб-застосунку

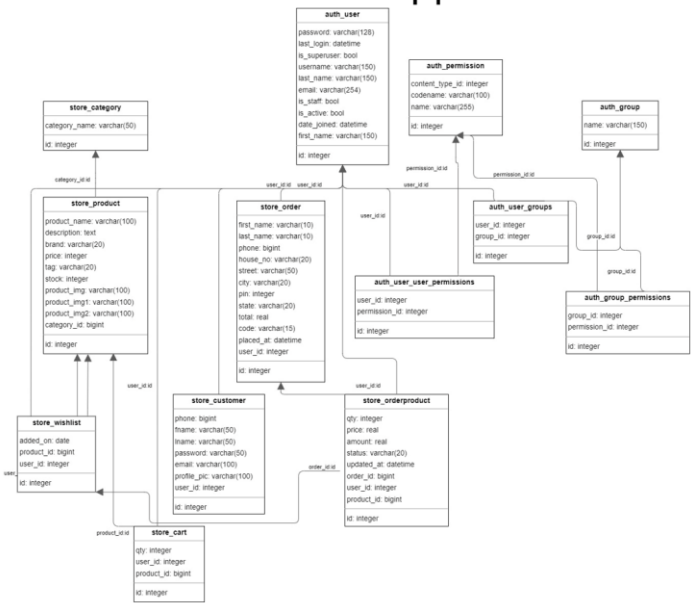
8

Діаграма класів



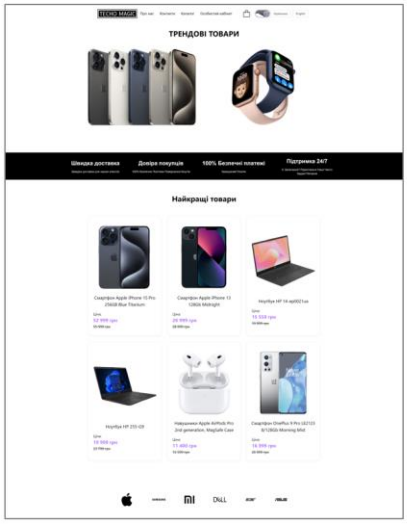
9

СХЕМА БАЗИ ДАНИХ



10

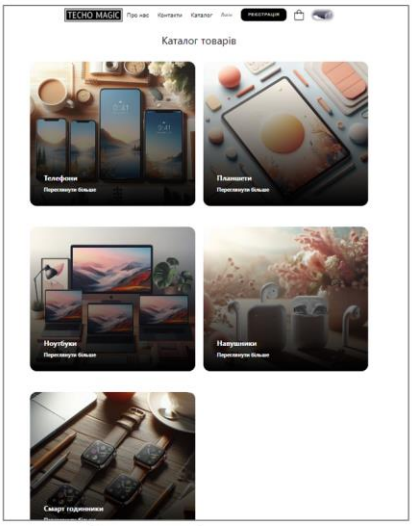
ЕКРАННІ ФОРМИ



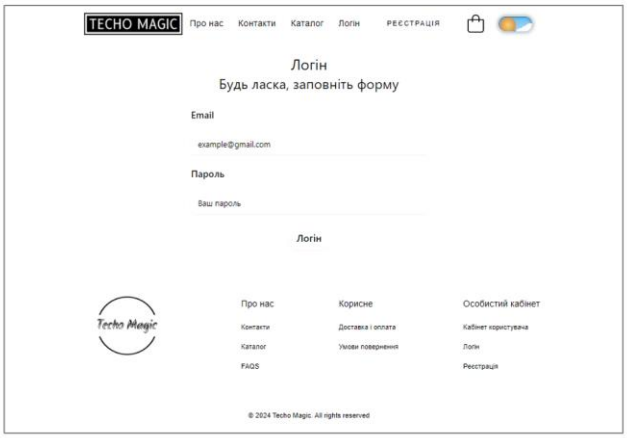
Головна сторінка каталогу web-застосунку

11

ЕКРАННІ ФОРМИ



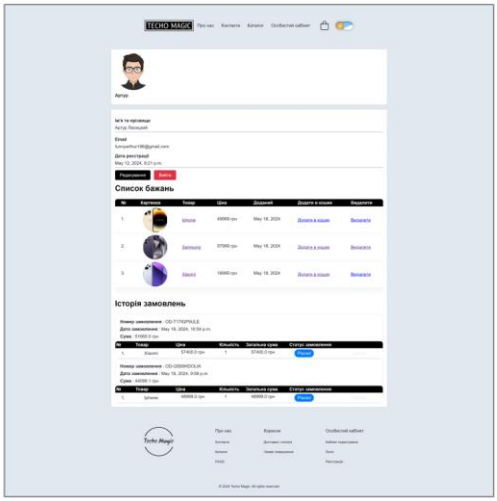
Каталог web-застосунку



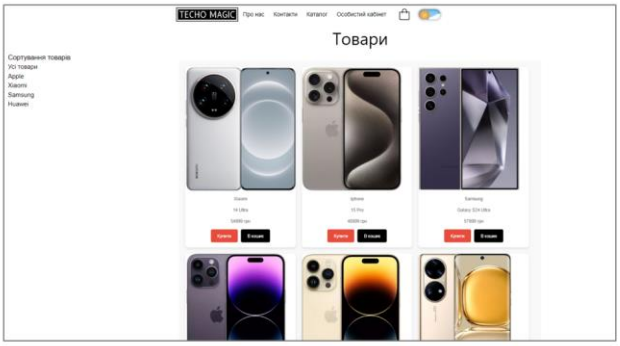
Логін web-застосунку

12

ЕКРАННІ ФОРМИ



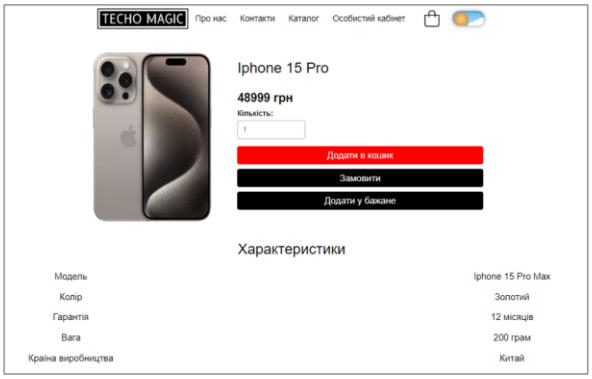
Особистий кабінет користувача



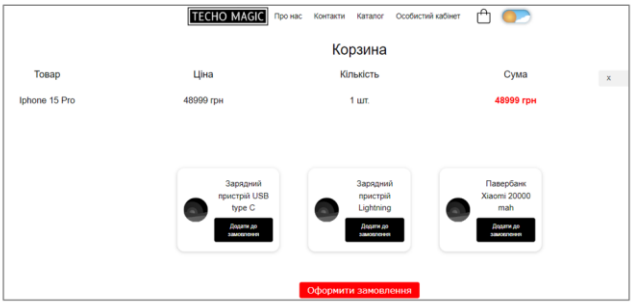
Каталог товарів

13

ЕКРАННІ ФОРМИ



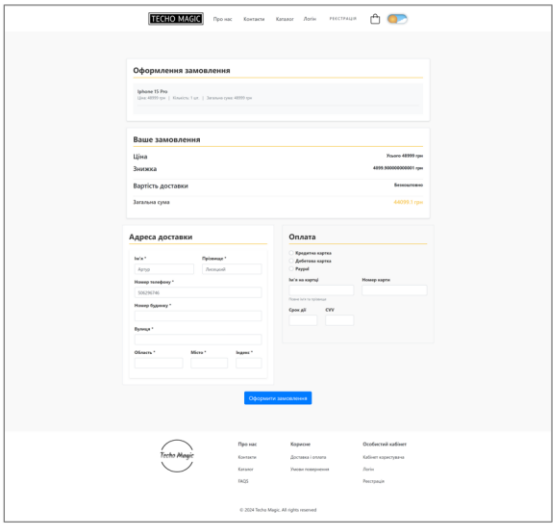
Сторінка товару



Кошик

14

ЕКРАННІ ФОРМИ



Оформлення замовлення

15

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Лісецький А.О.,Замрій І.В., ВИЗНАЧЕННЯ ВИМОГ ДО WEB-ЗАСТОСУНКУ З ПРОДАЖУ ТЕХНІКИ ІЗ ОНЛАЙН-СИСТЕМОЮ ОПЛАТИ. Застосування програмного забезпечення в ІКТ: Матеріали Всеукраїнська науково-технічна конференція. Збірник тез. 24.04.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024. С. 248-249.

2. Лісецький А.О.,Замрій І.В., БЕЗПЕКА WEB-ЗАСТОСУНКУ З ПРОДАЖУ ТЕХНІКИ З ОНЛАЙН-СИСТЕМОЮ ОПЛАТИ. Застосування програмного забезпечення в ІКТ: Матеріали Всеукраїнська науково-технічна конференція. Збірник тез. 24.04.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024. С. 253-254.

3. Лісецький А.О.,Замрій І.В., МОЖЛИВОСТІ ТА ПЕРЕВАГИ WEB-ЗАСТОСУНКУ ДЛЯ ПРОДАЖУ ТЕХНІКИ З ОНЛАЙН-СИСТЕМОЮ ОПЛАТИ. Застосування програмного забезпечення в ІКТ: Матеріали Всеукраїнська науково-технічна конференція. Збірник тез. 24.04.2024, ДУІКТ, м. Київ. К.: ДУІКТ, 2024. С.423-424

16

ВИСНОВКИ

1. Під час аналізу задач були визначені функціональні та нефункціональні вимоги до роботи веб-застосунку.
2. Було визначено ряд особливостей, відсутніх у існуючих веб-застосунків, такі як: можливість зміни мови інтерфейсу, можливість зміни теми інтерфейсу користувача, маленька кількість кроків для оформлення замовлення, що підкреслило унікальність розробки.
3. Проведено розробку архітектури системи, яка дозволила визначити, як повинна працювати система.
4. Проведено розробку клієнтської частини веб-застосунку, яка включала створення макету інтерфейсу, реалізацію інтерфейсу користувача.
5. Проведено розробку серверної частини веб-застосунку, яка включала створення бази даних, реалізацію функціональних вимог веб-застосунку.
6. Проведене тестування веб-застосунку за допомогою Unit-тестів підтвердило відсутність помилок при роботі веб-застосунку та надійну працездатність системи.

17

ДЯКУЮ ЗА УВАГУ!

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Models.py

```
from django.db import models
from django.contrib.auth.models import User
from django.utils.timezone import now

class Customer(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE, default="")
    phone = models.BigIntegerField()
    fname = models.CharField(max_length=50, default="")
    lname = models.CharField(max_length=50, default="")
    password = models.CharField(max_length=50, default="")
    email = models.CharField(max_length=100, default="")
    profile_pic = models.ImageField(null=True, upload_to="customer_pic", default="userimg.png")

    def __str__(self):
        return self.user.first_name + " " + self.user.last_name

class Category(models.Model):
    category_name = models.CharField(max_length=50)

    def __str__(self):
        return self.category_name

class Product(models.Model):
    product_name = models.CharField(max_length=100)
    category = models.ForeignKey(Category, on_delete=models.CASCADE)
    description = models.TextField(max_length=1000)
    brand = models.CharField(max_length=20, default="")
    price = models.IntegerField(default=0)
    tag = models.CharField(default="New", choices=TAG, max_length=20)
    stock = models.IntegerField(default=10)
    product_img = models.ImageField(upload_to="images", default="product.jpg")
    product_img1 = models.ImageField(upload_to="images", default="product.jpg")
    product_img2 = models.ImageField(upload_to="images", default="product.jpg")

    def __str__(self):
        return self.product_name

class Cart(models.Model):
    user = models.ForeignKey(User, on_delete=models.SET_NULL, null=True)
    product = models.ForeignKey(Product, on_delete=models.SET_NULL, null=True)
    qty = models.IntegerField()

    @property
    def price(self):
        return self.product.price

    @property
    def amount(self):
        return self.qty * self.product.price

    def __str__(self):
        return self.product.product_name + " by " + self.user.username
```

```
STATUS = (
    ('Placed', 'Placed'),
    ('Confirmed', 'Confirmed'),
    ('Preparing', 'Preparing'),
    ('Shipped', 'Shipped'),
    ('Out For Delivery', 'Out For Delivery'),
    ('Delivered', 'Delivered'),
    ('Cancelled', 'Cancelled'),
)

class Order(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    first_name = models.CharField(max_length=10)
    last_name = models.CharField(max_length=10)
    phone = models.BigIntegerField(null=True)
    house_no = models.CharField(null=True, max_length=20)
    street = models.CharField(null=True, max_length=50)
    city = models.CharField(null=True, max_length=20)
    pin = models.IntegerField(null=True)
    state = models.CharField(null=True, max_length=20)
    total = models.FloatField()
    code = models.CharField(max_length=15, default="")
    placed_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.code + " to " + self.first_name + "\t" + self.last_name

class OrderProduct(models.Model):
    order = models.ForeignKey(Order, on_delete=models.CASCADE)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    qty = models.IntegerField()
    price = models.FloatField()
    amount = models.FloatField()
    status = models.CharField(max_length=20, choices=STATUS, default='Placed')
    updated_at = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.product.product_name + " by " + self.user.first_name

class Wishlist(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    product = models.ForeignKey(Product, on_delete=models.CASCADE)
    added_on = models.DateField(default=now, editable=False)

    def __str__(self):
        return self.user.first_name + " - " + self.product.product_name
```

Products.py

```
from django.shortcuts import render
from store.models import Category, Customer, Product, Cart
def products(request):
    current_user = request.user
    customer = []
    try:
        customer = Customer.objects.get(user_id=current_user.id)
```

```
    except:
        pass
    categories = Category.objects.all()
    categoryid = 0
    filtered_category = [
    try:
        categoryid = request.GET['category']
    except:
        pass
    if categoryid:
        products = Product.objects.filter(category_id=categoryid)
        filtered_category = Category.objects.get(id=categoryid)
    else:
        products = Product.objects.all()
    n = len(products)
    current_user = request.user
    carts = Cart.objects.filter(user_id=current_user.id)
    qty = 0
    total = 0
    for cart in carts:
        total = total + cart.amount
        qty = qty + cart.qty

    params = {
        'customer': customer,
        'products': products,
        'categories': categories,
        'filtered_category': filtered_category,
        'n': n,
        'qty': qty,
        'total': total,
        'carts': carts,
    }
    return render(request, 'products.html', params)
```

Productdetail.py

```
from django.shortcuts import render
from store.models import Customer, Product, Cart, Wishlist


def productdetail(request, prid):
    current_user = request.user
    product = Product.objects.get(id=prid)
    carts = Cart.objects.filter(user_id=current_user.id)
    wishlist = Wishlist.objects.filter(user_id=current_user.id, product_id=prid)

    customer = []
    try:
        customer = Customer.objects.get(user_id=current_user.id)
    except:
        pass
    try:
        pr_qty = Cart.objects.get(user_id=current_user.id, product_id=prid)
        pr_qty = pr_qty.qty
    except:
        pr_qty = 0
```

```
    qty = 0
    total = 0
    for cart in carts:
        total = total + cart.amount
        qty = qty + cart.qty

    details = {
        'customer': customer,
        'product': product,
        'qty': qty,
        'total': total,
        'carts': carts,
        'wishlist': wishlist,
        'pr_qty': pr_qty
    }
    return render(request, 'product-item.html', details)
```

Makeorder.py

```
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from store.models import Customer, Order, OrderProduct, Cart
from django.contrib.auth.models import User
from django.shortcuts import redirect, render
from django.utils.crypto import get_random_string


@login_required(login_url='/login')
def checkout(request):
    currentuser = request.user
    carts = Cart.objects.filter(user_id=currentuser.id)
    customer = Customer.objects.get(user_id=currentuser.id)
    myuser = User.objects.get(id=currentuser.id)

    total = 0
    qty = 0
    for cart in carts:
        total = total + cart.amount
        qty = qty + cart.qty

    discount = 10 / 100 * total
    if discount > 20000:
        discount = 20000

    grand_total = total - discount

    if request.method == "POST":
        firstname = request.POST['firstname']
        lastname = request.POST['lastname']
        phone = request.POST['phone']
        house_no = request.POST['house_no']
        street = request.POST['street']
        state = request.POST['state']
        city = request.POST['city']
        pin = request.POST['pin']
        code = "OD-" + get_random_string(10).upper()

        order = Order(
```

```
            user_id=currentuser.id,
            first_name=firstname,
            last_name=lastname,
            phone=phone,
            house_no=house_no,
            street=street,
            city=city,
            pin=pin,
            state=state,
            total=grand_total,
            code=code
        )
        order.save()

        for cart in carts:
            orderpr = OrderProduct(
                order_id=order.id,
                user_id=currentuser.id,
                product_id=cart.product_id,
                qty=cart.qty,
                price=cart.product.price,
                amount=cart.amount
            )
            orderpr.save()

        Cart.objects.filter(user_id=currentuser.id).delete()
        messages.success(request, "Замовлення успішно оформлене!")
        return redirect('profile')

    details = {
        'myuser': myuser,
        'customer': customer,
        'carts': carts,
        'qty': qty,
        'total': total,
        'discount': discount,
        'grand_total': grand_total,
    }
    return render(request, 'make-order.html', details)
```

Buynow.py

```
from django.contrib.auth.decorators import login_required
from django.http.response import HttpResponseRedirect
from store.models import Cart

@login_required(login_url='/login')
def buynow(request, prid):
    current_user = request.user
    Cart.objects.filter(user_id=current_user.id).delete()
    data = Cart()
    data.user_id = current_user.id
    data.product_id = prid
    data.qty = 1
    data.save()
    return HttpResponseRedirect('/cart')
```

Login.py

```
from django.contrib import messages
from django.views import View
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login

class Login(View):
    def get(self, request):
        current_user = request.user
        if current_user.id:
            messages.success(request, 'Ви вже увійшли!')
            return redirect('profile')
        return render(request, 'login.html')

    def post(self, request):
        email = request.POST['email']
        password = request.POST['password']
        user = authenticate(request, username=email, password=password)
        if user is not None:
            login(request, user)
            return redirect('profile')
        else:
            messages.warning(request, "E-mail пароль не вірні!")
            return render(request, 'login.html', {'email':email})
```

Registration.py

```
from django.contrib import messages
from django.contrib.auth import authenticate, login
from django.views import View
from django.shortcuts import render, redirect
from django.contrib.auth.models import User
from store.models import Customer

class Registration(View):
    def get(self, request):
        current_user = request.user
        if current_user.id:
            messages.success(request, 'Ви вже увійшли!')
            return redirect('profile')
        return render(request, 'registration.html')

    def post(self, request):
        email = request.POST['email']
        fname = request.POST['fname']
        lname = request.POST['lname']
        phone = request.POST['phone']
        date = request.POST['date']
        pass1 = request.POST['pass1']
        uname = email
        print(uname, fname, lname, email, phone, pass1)

        values = {
            'email': email,
            'fname': fname,
            'lname': lname,
            'phone': phone,
            'date': date,
```

```
        }

        new_user = User.objects.create_user(
            uname,
            email=email,
            password=pass1,
            first_name=fname.capitalize(),
            last_name=lname.capitalize(),
        )
        new_user.save()

        customer = Customer(
            email=email,
            user=new_user,
            phone=phone,
            fname=fname,
            password=pass1,
            lname=lname,
        )
        customer.save()

        user = authenticate(request, username=email, password=pass1)
        if user is not None:
            login(request, user)

        messages.success(request, "Аккаунт зареєстровано!")
        return redirect('profile')
```

Profile.py

```
from store.models import Category, Customer, Order, OrderProduct, Cart, Wishlist
from django.contrib.auth.decorators import login_required
from django.shortcuts import render
from django.contrib.auth.models import User


@login_required(login_url='/login')
def account(request):
    orderprs = []
    currentuser = request.user

    carts = Cart.objects.filter(user_id=currentuser.id)
    qty = 0
    total = 0
    for cart in carts:
        total = total + cart.amount
        qty = qty + cart.qty

    categories = Category.objects.all()
    customer = Customer.objects.get(user_id=currentuser.id)
    wishlists = Wishlist.objects.filter(user_id=currentuser.id)

    orders = Order.objects.filter(user_id=currentuser.id).order_by("-placed_at")
    for order in orders:
        pr = OrderProduct.objects.filter(order_id=order.id)
        orderprs.append(pr)
```

```
    details = {
        'customer': customer,
        'orders': orders,
        'orderprs': orderprs,
        'qty': qty,
        'total': total,
        'carts': carts,
        'categories': categories,
        'wishlists': wishlists
    }

    return render(request, 'profile.html', details)
```

Views.py

```
from django.shortcuts import render
def login(request):
    return render(request, 'login.html')

def registration(request):
    return render(request, 'registration.html')

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

def products(request):
    return render(request, 'products.html')

def makeOrder(request):
    return render(request, 'make-order.html')

def profile(request):
    return render(request, 'profile.html')

def updateprofile(request):
    return render(request, 'updateProfil
e.html')
```

Wishlist.py

```
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from store.models import Product, Wishlist
from django.http.response import HttpResponseRedirect

@login_required(login_url='/login')
def addtowishlist(request, prid):
    url = request.META.get('HTTP_REFERER')
    current_user = request.user
    wishlist = Wishlist(
        user_id = current_user.id,
        product_id = prid
    )
    wishlist.save()
    messages.success(request, wishlist.product.product_name + " added to your Wishlist.")
    return HttpResponseRedirect(url)

@login_required(login_url='/login')
def removefromwishlist(request, prid):
    url = request.META.get('HTTP_REFERER')
```

```
    current_user = request.user
    product = Product.objects.get(id=prid)
    Wishlist.objects.get(user_id = current_user.id, product_id = prid).delete()
    messages.success(request, product.product_name + " removed to your Wishlist.")
    return HttpResponseRedirect(url)
```

Cart.py

```
from store.models import Customer, Cart
from django.contrib.auth.decorators import login_required
from django.shortcuts import render

@login_required(login_url='/login')
def cart(request):
    current_user = request.user
    customer = Customer.objects.get(user_id=current_user.id)
    carts = Cart.objects.filter(user_id=current_user.id)

    total = 0
    qty = 0
    for cart in carts:
        total = total + cart.amount
        qty = qty + cart.qty

    discount = 10 / 100 * total
    if discount > 20000:
        discount = 20000
    grand_total = total - discount
    cart = {
        'customer': customer,
        'carts': carts,
        'qty': qty,
        'total': total,
        'discount': discount,
        'grand_total': grand_total,
    }
    return render(request, 'cart.html', cart)
```

Cartoperations.py

```
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from store.models import Cart
from django.http.response import HttpResponseRedirect

@login_required(login_url='/login')
def addtocart(request, prid):
    url = request.META.get('HTTP_REFERER')
    current_user = request.user
    check_product = Cart.objects.filter(user_id=current_user.id, product_id=prid)

    if check_product:
        control = 1
    else:
        control = 0

    if control == 1:
        data = Cart.objects.get(user_id=current_user.id, product_id=prid)
        data.qty = data.qty + 1
        data.save()
```

```
    else:
        data = Cart()
        data.user_id = current_user.id
        data.product_id = prid
        data.qty = 1
        data.save()
    messages.success(request, data.product.product_name + " added to the Cart.")
    return HttpResponseRedirect(url)

def deletefromcart(request, prid):
    url = request.META.get('HTTP_REFERER')
    current_user = request.user
    product = Cart.objects.get(user_id=current_user.id, product_id=prid)
    if product.qty == 1:
        product.delete()
    else:
        product.qty = product.qty - 1
        product.save()
    messages.success(request, product.product.product_name + " deleted from the Cart.")
    return HttpResponseRedirect(url)

def deleteallfromcart(request, prid):
    current_user = request.user
    Cart.objects.get(product_id=prid, user_id=current_user.id).delete()
    return HttpResponseRedirect('/cart')

def clearcart(request):
    current_user = request.user
    Cart.objects.filter(user_id=current_user.id).delete()
    return HttpResponseRedirect('/cart')
```

Urls.py

```
from django.contrib import admin
from django.urls import path
from store.login import Login
from store.registration import Registration
from store.logout import Logout
from store.buynow import buynow
from store.cart import cart
from store.productdetail import productdetail
from store.products import products
from store.makeOrder import checkout
from store.profile import account
from store.wishlist import addtowishlist, removefromwishlist
from store.cartoperations import clearcart, addtocart, deletefromcart, deleteallfromcart
from store.views import updateprofile

urlpatterns = [
    path('admin/', admin.site.urls),
    path('login/', Login.as_view(), name='login'),
    path('registration/', Registration.as_view(), name='registration'),
    path('cart/', cart, name='cart'),
    path('products/', products, name='products'),
    path("productdetail/<int:prid>", productdetail, name="ProductDetail"),
    path('makeOrder/', checkout, name='makeOrder'),
    path('profile/', account, name='profile'),
    path("profile/updateprofile/", updateprofile, name="updateProfile"),
```

```
    path('logout/', Logout, name='logout'),
    path("buynow/<int:prid>", buynow, name="BuyNow"),
    path("clearcart/", clearcart, name="ClearCart"),
    path("deletefromcart/<int:prid>", deletefromcart, name="DeletefromCart"),
    path("deleteallfromcart/<int:prid>", deleteallfromcart, name="DeleteAllfromCart"),
    path("addtocart/<int:prid>", addtocart, name="AddtoCart"),
    path("addtowishlist/<int:prid>", addtowishlist, name="AddfromWishlist"),
    path("removefromwishlist/<int:prid>", removefromwishlist, name="RemovefromWishlist"),
]
```