Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»

РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ ТЕХНІЧНИХ ТОВАРІВ НА МОВІ PYTHON

DEVELOPMENT OF AN ONLINE STORE OF TECHNICAL PRODUCTS IN PYTHON PROGRAMMING LANGUAGE

спеціальність 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

освітня програма «Інженерія програмного забезпечення»
(назва освітньої програми)

Виконав: здобувач вищої освіти
Групи ІПЗс-31
Вовчук Матвій Олександрович
_____
(підпис)

Керівник:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____
(підпис)

Кваліфікаційну роботу
допущено до захисту
«8» 06 2023 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна
_____
(підпис)

Луцьк – 2023 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра інженерії програмного забезпечення
Ступінь вищої освіти: *бакалавр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *121 Інженерія програмного забезпечення*
Освітня програма: *«Інженерія програмного забезпечення»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
«2» 02 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Вовчук Матвій Олександрович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Розробка інтернет-магазину технічних товарів на мові Python

Керівник роботи: д.т.н., доцент Ліщина Наталія Миколаївна
затверджені наказом закладу вищої освіти від «*23*» *грудня 2022* р. № *961-01-02*

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи *«8» червня 2023 р.*

3. Вихідні дані до роботи: *технічне та програмне забезпечення ЕОМ, вимоги до розробки програмного забезпечення, ергономічні вимоги до функціонування програмного засобу, Мова програмування серверної частини – Python, Технології програмування клієнтської частини – HTML, CSS, JavaScript, AJAX.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити):
Аналіз сучасного стану проблеми, існуючих методів і засобів її розв'язання, аналіз і вибір засобів проектування, опис функціонального наповнення об'єкта проектування, розробка й обґрунтування системного наповнення, оцінка ергономічних та надійністних параметрів проектованої системи, функціонально-структурна схема роботи об'єкта проектування, розробка моделі бази даних об'єкта проектування.

5. Перелік графічного матеріалу: 20 рис; 4 схеми; 3 діаграм.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	*Ліщина Н.М.*		
Специфікація вимог до розробленої системи	*Ліщина Н.М.*		
Розробка об'єкта проектування	*Ліщина Н.М.*		
Нормоконтроль	*Ліщина Н.М.*		
Гарант ОП	*Ліщина Н.М.*		
Показник запозичень тексту	13,4%		
Академічна доброчесність	*Ящук А.А.*		

7. Дата видачі завдання *«2» лютого 2023 р.*

КАЛЕНДАРНИЙ ПЛАН

№ З/п	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітки
1	*Огляд літературних джерел по темі кваліфікаційної роботи бакалавра*	*05.02.2023 р.*	
2	*Аналіз проблеми розробки та впровадження об'єкту проектування*	*27.02.2023 р.*	
3	*Обґрунтування вибору шляхів, технологій (алгоритмів) і засобів вирішення поставленого завдання*	*10.03.2023 р.*	
4	*Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних*	*17.04.2023 р.*	
5	*Практична реалізація об'єкта проектування та розробка бази даних*	*18.05.2023 р.*	
6	*Тестування та налагодження об'єкта проектування*	*01.06.2023*	
7	*Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедру*	*08.06.2023 р.*	

Здобувач вищої освіти _____ Вовчук М.О.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Н. Ліщина
(підпис) (прізвище, ініціали

Міністерство освіти і науки України
Луцький національний технічний університет

Рецензія
на кваліфікаційну роботу

Здобувач вищої освіти:	Вовчук Матвій Олександрович
Спеціальність і група:	121 Інженерія програмного забезпечення, ІПЗс-31
Обсяг кваліфікаційної роботи бакалавра:	72 стор., 20 рис., 17 літературних джерел
Кількість сторінок записки:	72 сторінка

Тема: ***Розробка інтернет-магазину технічних товарів на мові Python***

Коротка характеристика кваліфікаційної роботи та прийнятих рішень:
Кваліфікаційна робота бакалавра складається з трьох розділів, в яких проаналізовані проблематика та поставлені завдання за темою роботи, здійснено теоретичне дослідження та практичну реалізацію інтернет-магазину технічних товарів на мові Python.

Самостійні розробки і пропозицій автора: *Кваліфікаційна робота бакалавра присвячена розробці програмного продукту, який буде використовуватись для продажу технічних товарів. Користувачу представлений комфортний та функціональний інтерфес .для оформлення замовлень.*

Недоліки: *Істотних недоліків в роботі не виявлено.*

Загальний висновок: *У кваліфікаційній роботі бакалавра було спроектовано та створено веб застосунок для продажу технічних товарів. Робота є результатом самостійного дослідження. Істотних недоліків не виявлено. Кваліфікаційна робота здобувача освіти Вовчука Матвія рекомендується до захисту екзаменаційній комісії та заслуговує відмінної оцінки.*

Рецензент: Гуцалюк Юрій Йосипович д. пед. н.
(прізвище, ім'я, по-батькові, посада)

Рецензент кваліфікаційної роботи _____
(підпис)

«08» червня 2023 р.

АНОТАЦІЯ

Вовчук М.О. Розробка інтернет-магазину технічних товарів на мові Python. Рукопис.

Кваліфікаційна робота бакалавра за спеціальністю 121 «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2023. 72 с.

Кваліфікаційна робота присвячена розробці інтернет-магазину технічних товарів на мові Python.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків.

У першому розділі проведено аналіз сучасного стану проблеми і сформульовано завдання для кваліфікаційної роботи бакалавра.

В другому розділі встановлено вимоги до розроблюваної системи та розроблено програмне забезпечення. Вибрано засоби, методи і алгоритми для вирішення завдань.

Третій розділ присвячено розробці веб-сайту для продажу технічних товарів з використанням фреймворку Django. Розглянуто реалізацію об'єкта проектування, розроблено базу даних, проведено тестування та налагодження системи. Також створено документацію для програмного продукту і інсталяційний пакет.

У висновках роботи підводяться загальні підсумки щодо досягнутих результатів. Крім того, наводиться список використаних джерел і додаються додатки, які доповнюють основний зміст роботи.

Ключові слова: інтернет-магазин, веб-додаток, JavaScript, Python, Django, HTML, CSS, SASS, Database.

ANNOTATION

Vovchuk M.O. Development of an online store of technical products in Python. Manuscript.

Bachelor's thesis in the specialty 121 "Software Engineering". Lutsk National Technical University. Lutsk, 2023. 72 p.

The qualification work is devoted to the development of an online store of technical products in Python.

The bachelor's thesis consists of an introduction, three chapters, conclusions and suggestions, a list of references and appendices.

The first section analyzes the current state of the problem and formulates the task for the bachelor's thesis.

The second section sets out the requirements for the system to be developed and develops the software. The tools, methods and algorithms for solving the tasks are selected.

The third section is devoted to the development of a website for the sale of technical goods using the Django framework. The realization of the design object is considered, a database is developed, and the system is tested and debugged. The documentation for the software product and the installation package were also created.

The conclusion of the paper summarizes the general results achieved. In addition, a list of references is provided and appendices are attached to supplement the main content of the paper.

Keywords: online store, web application, JavaScript, Python, Django, HTML, CSS, SASS, Database.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі електронна комерція є однією з найбільш перспективних та швидкорозвиваючихся галузей бізнесу. За останні роки, зростання інтернет-магазинів набуло неабиякої актуальності в контексті електронної комерції. У зв’язку з цим, розробка та підтримка інтернет-магазину стає ключовим елементом успішної бізнес-стратегії для багатьох компаній.

Однією з основних переваг інтернет-магазинів з технічними товарами є можливість покупки товарів з будь-якої точки світу, що дозволяє збільшити аудиторію клієнтів та збільшити продажі. Крім того, інтернет-магазини забезпечують широкий асортимент товарів та можливість порівняння цін, що дозволяє клієнтам знайти оптимальний варіант для себе.

Важливо звернути увагу на питання безпеки придбання товарів в інтернет-магазинах з технічними товарами. Клієнт повинен бути впевнений у тому, що його персональна інформація та оплата за товар будуть захищені від шахраїв та шкідливих дій третіх осіб.

Також, інтернет-магазини з технічними товарами є дуже зручними для бізнесу, який використовує техніку та електроніку. Оскільки у бізнесу часто потрібні різні технічні товари, інтернет-магазини з технічними товарами дозволяють швидко та зручно замовляти необхідне обладнання та інші технічні товари для компанії.

Крім того, у зв'язку з пандемією COVID-19 та зростанням онлайн-шопінгу, розвиток інтернет-магазинів став ще актуальнішим, оскільки вони дозволяють клієнтам купувати товари безпечно та зручно з власного дому.

Тому, розробка інтернет-магазину технічних товарів на мові програмування Python є актуальною темою для кваліфікаційної роботи. У зв’язку з швидким розвитком технологій, існує потреба в створенні нових та вдосконаленні існуючих програмних продуктів, які забезпечують високу якість обслуговування клієнтів та ефективне управління бізнесом.

Об'єкт дослідження даної кваліфікаційної роботи – розробка інтернет-магазину технічних товарів на мові програмування Python. Дослідження включає розробку програмного забезпечення, використання сучасних технологій та бібліотек для досягнення найкращої продуктивності та забезпечення користувачам максимального комфорту під час використання додатку.

Предмет дослідження є аналіз існуючих технологій та інструментів, які використовуються при розробці інтернет-магазину на мові Python, створення прототипу магазину, визначення його функціональних можливостей та технічних вимог.

Метою дослідження є розробка програмного забезпечення функціонального інтернет-магазину технічних товарів на мові програмування Python з використанням сучасних технологій та інструментів включаючи фреймворк Django.

Завдання дослідження полягають у вивченні сучасних технологій та інструментів для розробки інтернет-магазину на мові Python, проектуванні структури бази даних, розробці відповідного функціоналу, який дозволить користувачам здійснювати замовлення та оплату товарів, побудові системи безпеки та захисту даних клієнтів, а також тестуванні та випробуванні розробленого інтернет-магазину.

Новизна роботи. Новизною даної роботи є розробка функціонального інтернет-магазину технічних товарів на мові програмування Python з використанням сучасних технологій та інструментів, що відповідає вимогам сучасного електронного бізнесу. Крім того, у цій роботі будуть враховані особливості розробки магазину на мові програмування Python, що дозволить вирішити проблеми, які можуть виникнути під час розробки цього типу програмного забезпечення.

У світі інтернет-магазини є ключовим елементом електронної комерції. Тому, розробка функціонального інтернет-магазину технічних товарів на мові програмування Python є дуже актуальною та перспективною темою для

дослідження. Результати даної роботи можуть бути корисними для розробників програмного забезпечення, бізнесу та користувачів, які бажають зайнятися електронною комерцією.

Кваліфікаційна робота пройшла апробацію на Міжнародній науково-практичній конференції з проблем вищої освіти і науки «Інформаційні технології в освіті, науці і виробництві (ІТОНВ-2023)». Тема доповіді: "Розробка ефективної системи управління контентом для інтернет-магазину на основі DJANGO".

РОЗДІЛ 1
АНАЛІЗ ІНТЕРНЕТ-МАГАЗИНІВ ТЕХНІЧНИХ ТОВАРІВ І ПОСТАНОВКА ЗАВДАНЬ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

На сьогоднішній день, зростає популярність інтернет-магазинів з технічними товарами, оскільки вони забезпечують зручний та швидкий доступ до великої кількості товарів з різних галузей техніки та електроніки, та являються швидким способом придбання необхідних речей для дому, офісу чи особистого використання.

Однією з основних переваг інтернет-магазинів з технічними товарами є можливість покупки товарів з будь-якої точки світу, що дозволяє збільшити аудиторію клієнтів та збільшити продажі. Крім того, інтернет-магазини забезпечують широкий асортимент товарів та можливість порівняння цін, що дозволяє клієнтам знайти оптимальний варіант для себе.

Важливо звернути увагу на питання безпеки придбання товарів в інтернет-магазинах з технічними товарами. Клієнт повинен бути впевнений у тому, що його персональна інформація та оплата за товар будуть захищені від шахраїв та шкідливих дій третіх осіб.

Також, інтернет-магазини з технічними товарами є дуже зручними для бізнесу, який використовує техніку та електроніку. Оскільки у бізнесу часто потрібні різні технічні товари, інтернет-магазини з технічними товарами дозволяють швидко та зручно замовляти необхідне обладнання та інші технічні товари для компанії.

Крім того, у зв'язку з пандемією COVID-19 та зростанням онлайн-шопінгу, розвиток інтернет-магазинів став ще актуальнішим, оскільки вони дозволяють клієнтам купувати товари безпечно та зручно з власного дому.

У цілому, необхідність у потребі інтернет-магазинів з технічними товарами є актуальною та зростаючою, проте для успішної роботи в даній галузі необхідне

постійне вдосконалення сервісу та розвиток у напрямку забезпечення якості обслуговування та безпеки покупців. Необхідно забезпечити простоту та зручність вибору товару, надати достатньо інформації та консультацій клієнтам. Також, важливо створити високий рівень довіри до магазину та гарантувати безпеку оплати за товар.

Для розробки веб ресурсу для продажу технічних товарів потрібно провести аналіз серед аналогів, щоб уникнути недоліків, яких допустили конкуренти. Для аналізу існуючих аналогів було обрано наступні веб ресурси:

- Rozetka;
- Allo;
- Amazon;
- Telemart.ua.

Інтернет-магазин Rozetka – це один з найбільших інтернет-ритейлерів в Україні, який був заснований в 2005 році. Магазин пропонує широкий асортимент товарів, включаючи електроніку, побутову техніку, товари для дому та саду, одяг та взуття, косметику та парфумерію, дитячі товари та багато іншого (рис. 1.1, 1.2, 1.3).

У 2021 році компанія була визнана найбільш успішним онлайн-магазином в Україні за рівнем впливу та проникнення на ринок, за даними дослідження інституту Retail Rocket. Статистика показує, що на кінець 2021 року Rozetka мала понад 14 мільйонів зареєстрованих користувачів та більше 1,2 мільйонів замовлень на місяць.

Переваги інтернет-магазину Rozetka:

- широкий асортимент товарів – магазин пропонує велику кількість технічних товарів, включаючи комп'ютери, ноутбуки, смартфони, телевізори, побутову техніку та багато іншого;
- зручний та простий інтерфейс – сайт має зрозумілий та добре організований інтерфейс, який дозволяє легко знайти потрібний товар та здійснити покупку;

– швидка доставка – магазин має велику мережу доставки по всій Україні та надає можливість швидкої доставки замовлення, що є великою перевагою для клієнтів;

– гарантія на товар – магазин надає гарантію на товари, що дає клієнтам впевненість у якості та надійності куплених товарів;

– клієнтська підтримка – магазин має професійну та допоміжну службу підтримки, яка готова допомогти з будь-якими питаннями чи проблемами.

Недоліками магазину є:

– високі ціни – на деякі товари можуть бути високі ціни, порівняно з іншими магазинами;

– недоступність товарів – на деякі товари може бути довгий термін очікування, оскільки вони не завжди є в наявності;

– проблеми з поверненням товарів – деякі покупці зазначають проблеми з поверненням товарів та отриманням повернення коштів, що може вплинути на репутацію магазину та знизити довіру клієнтів;

– невідповідність товарів – деякі покупці зазначають, що отримані товари не відповідають опису на сайті.

Рисунок 1.1 – Пункт видачі товарів магазину

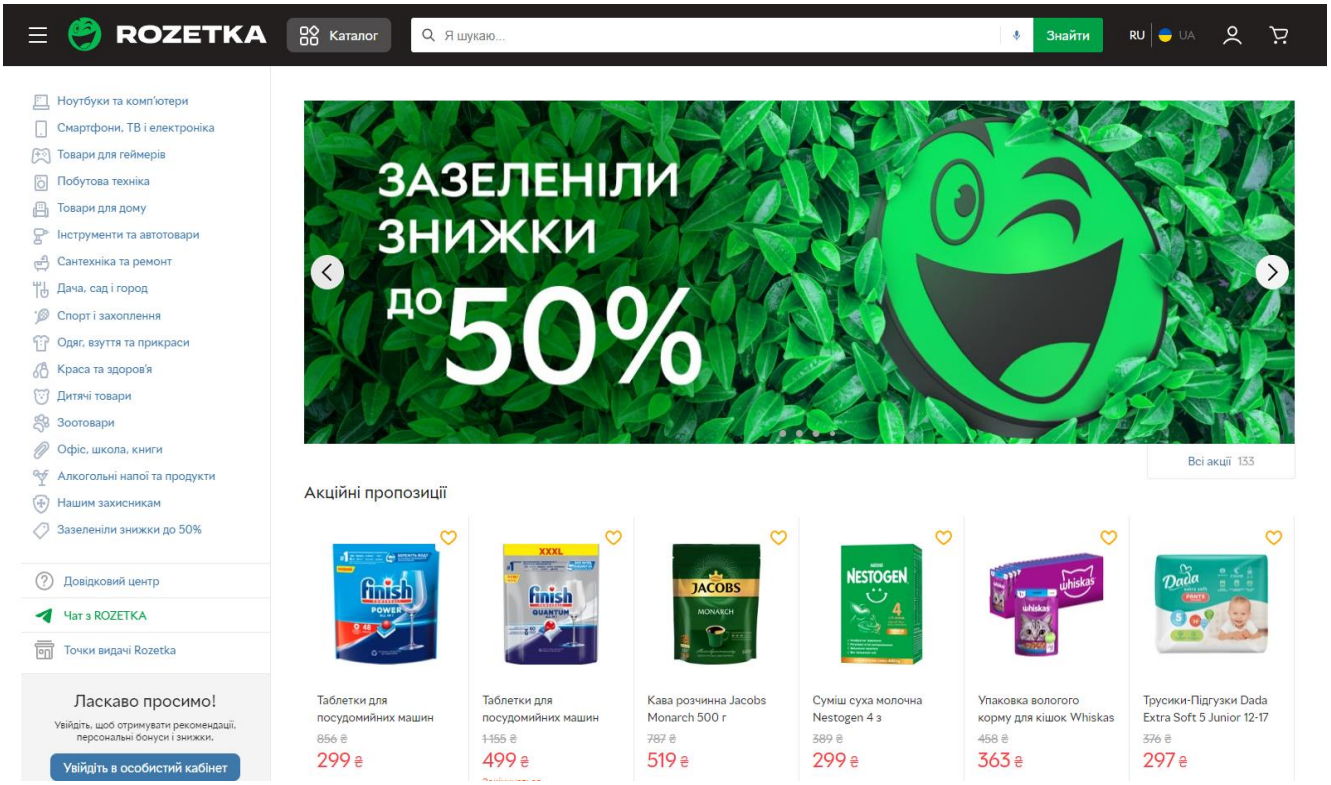

Рисунок 1.2 – Інтерфейс інтернет-магазину

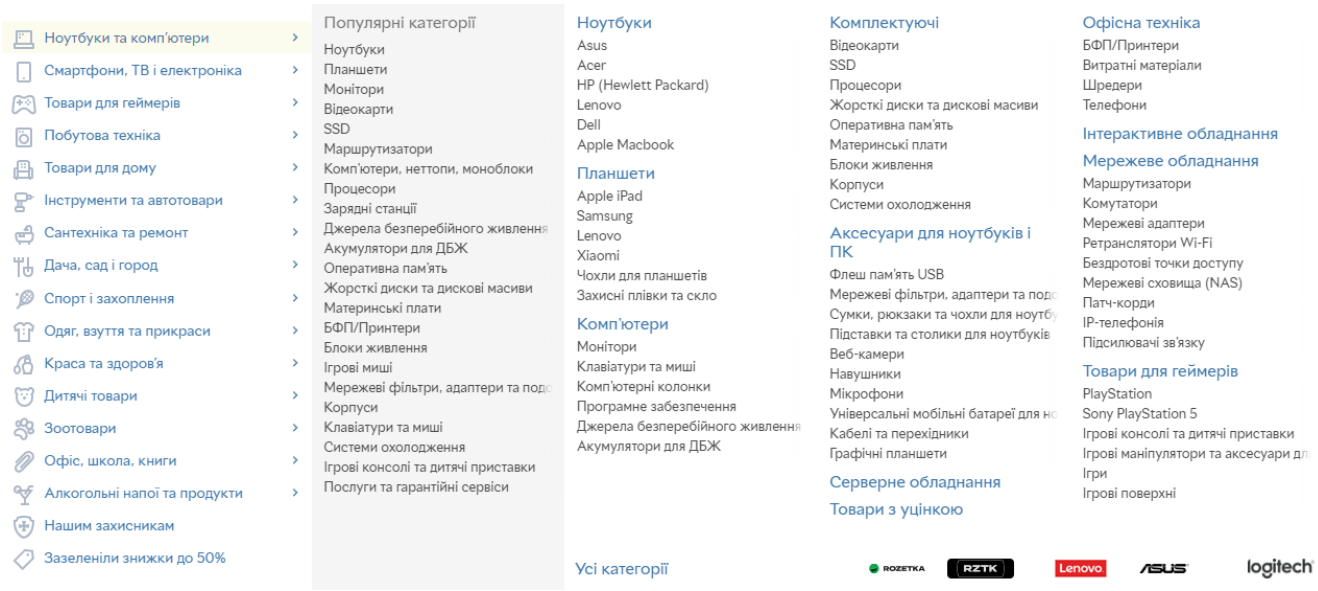

Рисунок 1.3 – Великий каталог товарів або меню магазину

Інтернет-магазин «Allo.ua» – це один з найбільших українських онлайн-роздрібників, що пропонує широкий асортимент електроніки, побутової техніки та інших товарів для дому. Заснований у 2000 році, магазин став лідером у своїй галузі завдяки широкому асортименту, конкурентним цінам та якісному обслуговуванню.

Згідно з даними SimilarWeb, на березень 2021 року, серед українських інтернет-магазинів "Allo.ua" займає перше місце за кількістю відвідувань, які складають більше 24 млн. на місяць. Також, відповідно до даних EVO-Rating, в 2020 році "Allo.ua" посів перше місце серед інтернет-магазинів за обсягом продажів в Україні, що склав 7,5 млрд гривень.

Переваги магазину Allo:

– широкий асортимент технічних товарів – можна знайти багато різних категорій технічних товарів, що дозволяє задовольнити потреби різних груп покупців;

– низькі ціни – порівняно з іншими інтернет-магазинами, Allo пропонує конкурентоспроможні ціни на багато товарів;

– зручність – інтерфейс сайту простий та зручний, що дозволяє з легкістю знайти потрібний товар та зробити замовлення;

– висока якість обслуговування – компанія пропонує швидку доставку, а також можливість повернення товару протягом 14 днів;

– програма лояльності – для постійних покупців доступна програма лояльності, що дозволяє отримувати додаткові знижки та бонуси за покупки (рис. 1.4, 1.5).

Недоліками магазину є:

– проблеми з доставкою – у деяких регіонах України можуть виникати проблеми з доставкою товару, що може призвести до затримок та нестабільності сервісу;

– відсутність опису товарів – у деяких випадках на сайті відсутній детальний опис товару, що може ускладнити вибір покупця;

– відсутність програми лояльності: магазин не має програми лояльності, яка б дозволяла накопичувати бонуси або знижки при покупках;

– відсутність гарантії на товари – компанія не завжди надає гарантію на товари, що може стати проблемою для покупців у випадку зіпсованого товару;

– обмежена географія – компанія не працює з усіма регіонами України, що обмежує географію своєї аудиторії та можливості збільшення продажів;

– відсутність можливості оплати готівкою: магазин не надає можливість оплати готівкою при отриманні товару, що може відлякувати деяких покупців;

– відсутність інформації про наявність товарів: інформація про наявність товарів на сайті може бути неповною, що може створювати незручності при замовленні.

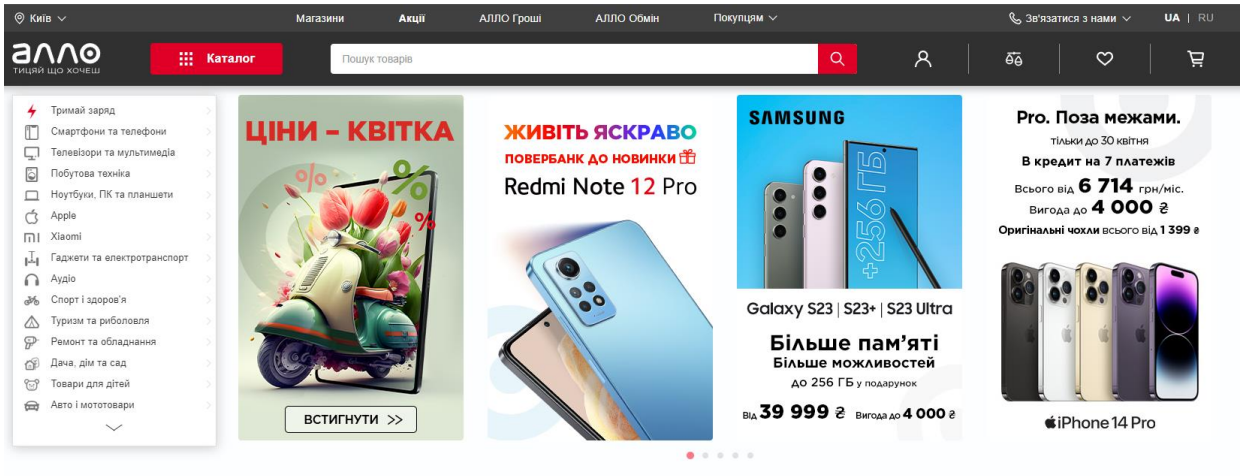

Рисунок 1.4 – Інтерфейс головної сторінки магазину Allo

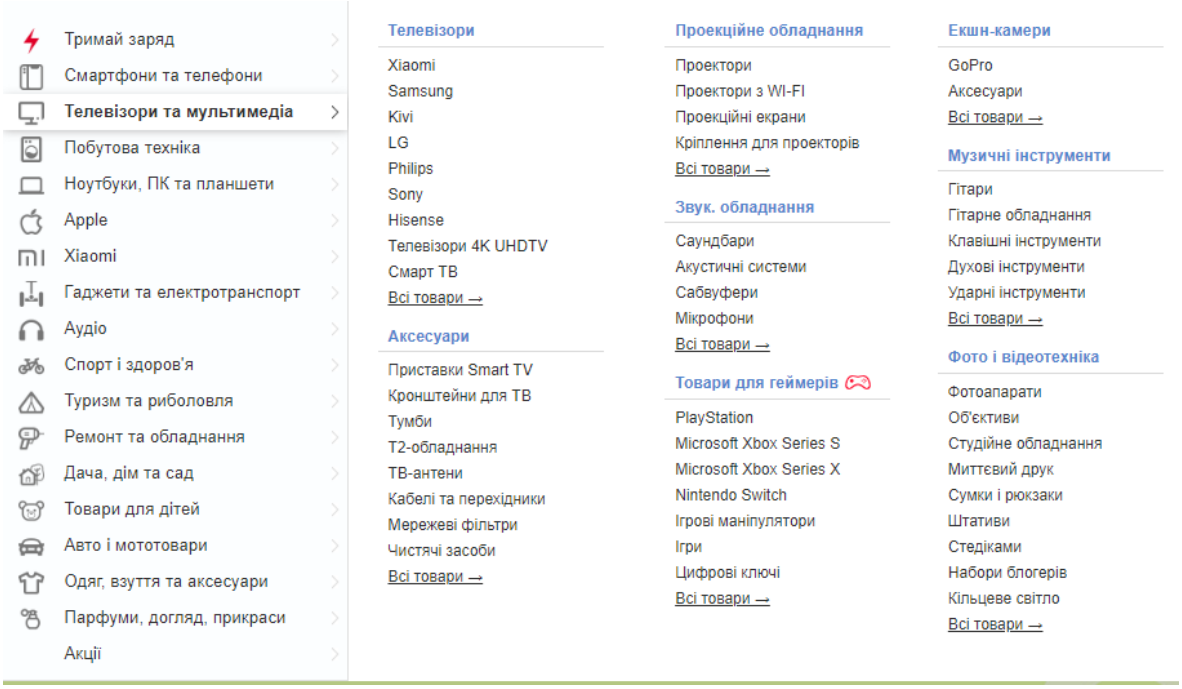

Рисунок 1.5 – Каталог товарів магазину Allo

Amazon – це один з найбільших інтернет-магазинів у світі, який пропонує широкий вибір товарів різних категорій, включаючи електроніку, книги, одяг, косметику, продукти харчування та багато іншого (рис. 1.6). Компанія Amazon була заснована у 1994 році Джефом Безосом, а в 1995 році запустила свій перший веб-сайт, який почав продавати книги онлайн. З тих пір Amazon значно розширив свій асортимент товарів та послуг, а також став одним з найпопулярніших онлайн-магазинів у світі.

Сьогодні Amazon має велику базу клієнтів по всьому світу, що складається з понад 300 мільйонів активних користувачів. Компанія також запустила свій власний бренд Amazon Basics, який пропонує доступні варіанти електроніки, одягу та інших товарів.

Переваги:

– великий вибір товарів: Amazon має величезний асортимент товарів з різних категорій, що дозволяє привернути до себе широку аудиторію;

– міжнародна доставка: Amazon пропонує доставку в різні країни світу, що дозволяє покупцям з різних країн замовляти товари;

– розширена система гарантій: Amazon пропонує різні типи гарантій на свої товари, що дозволяє залучати більше покупців;

– відгуки покупців: Amazon дозволяє покупцям залишати відгуки про товари, що дозволяє іншим покупцям приймати рішення про купівлю;

– програма підписки Amazon Prime: ця програма дозволяє покупцям отримувати безкоштовну доставку, спеціальні пропозиції та інші переваги.

Недоліки:

– складний інтерфейс сайту: інтерфейс Amazon може бути складним для користувачів, що може призводити до складнощів у пошуку та замовленні товарів;

– проблеми зі здійсненням оплати: Amazon пропонує кілька варіантів оплати, але деякі методи оплати можуть бути недоступні для деяких країн;

– приховані вартості: деякі покупці можуть не задоволені тим, що деякі вартості (наприклад, податки) не включені в ціну товару і можуть додатково сплачувати за це (рис. 1.6).

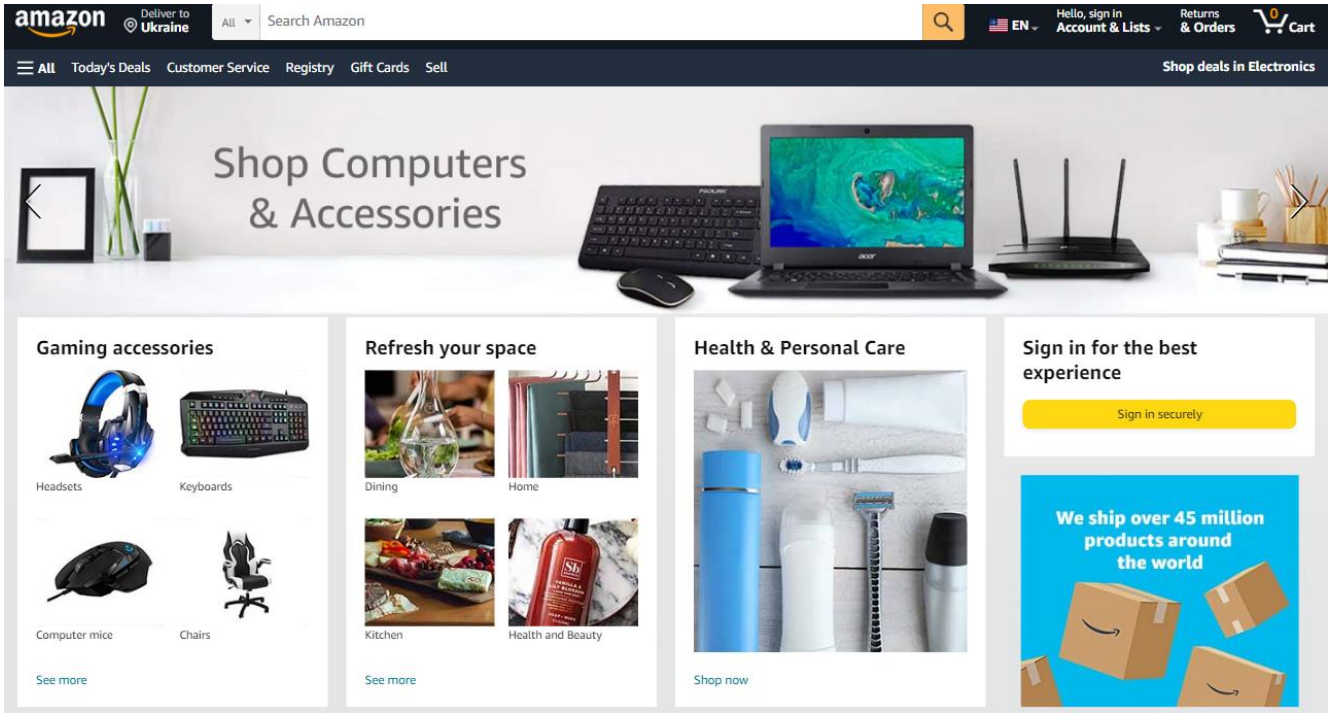

Рисунок 1.6 – Інтерфейс головної сторінки магазину Amazon

Telemart.ua – це один з найбільших інтернет-магазинів техніки та електроніки в Україні. Магазин пропонує великий асортимент товарів, таких як смартфони, планшети, ноутбуки, комп'ютери, телевізори, фото та відеотехніка, побутова техніка та багато іншого (рис. 1.7).

Переваги:

– широкий вибір товарів: telemart.ua має великий асортимент технічних товарів, включаючи смартфони, ноутбуки, телевізори, планшети та інші товари. Це дозволяє задовольняти потреби різних клієнтів та забезпечувати більш високу конкурентоспроможність;

– зручний та простий інтерфейс: інтерфейс telemart.ua простий та легкий у використанні, що дозволяє користувачам швидко знайти та замовити потрібні товари;

– швидка доставка: магазин пропонує швидку доставку товарів, що дозволяє клієнтам отримати їх у найкоротші терміни;

– конкурентоспроможні ціни: telemart.ua пропонує товари за конкурентоспроможними цінами, що дозволяє приваблювати нових клієнтів та зберігати існуючих;

– конфігуратор компонентів ПК: на сайті можна підібрати сумісні до інших компонентів комплектуючі.

Недоліки:

– обмежена гарантія: у telemart.ua надається обмежена гарантія на товари, що може знизити довіру клієнтів до магазину;

– відсутність можливості повернення товару: магазин не надає можливості повернення товару за відсутності вад або недоліків;

– обмежена опція оплати: telemart.ua пропонує обмежену кількість опцій оплати, що може бути не зручним для деяких клієнтів;

– не завжди доступні товари: на деякі товари у telemart.ua може бути низька наявність, що може викликати незадоволення клієнтів.

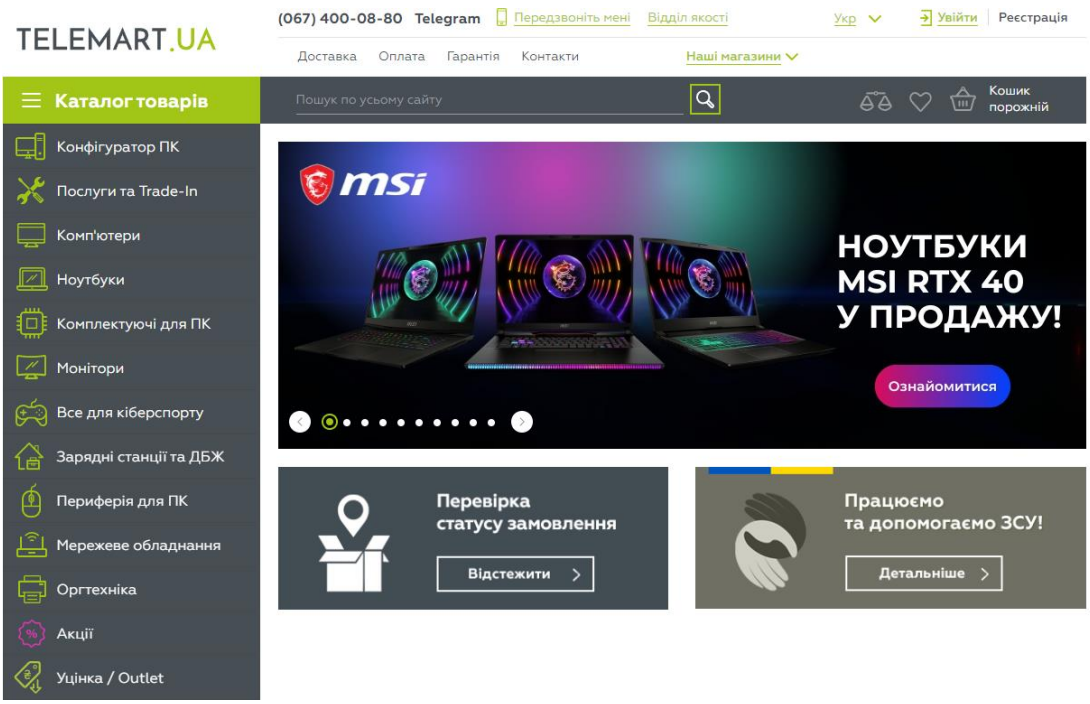

Рисунок 1.7 – Інтерфейс головної сторінки магазину Telemart.ua

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

Метою даної роботи є створення інтернет-магазину технічних товарів, який буде ефективним інструментом продажу товарів та послуг на ринку електронної комерції. Клієнти матимуть можливість переглядати: каталог з усіма товарами, відгуки інших користувачів, опис товару, статус відправки товару. Матимуть можливість залишити свій відгук, після завершення замовлення. Після реєстрації користувач матиме власний обліковий запис із його особистими даними.

Для досягнення кінцевої мети необхідно виконати наступний список завдань:

– проведення аналізу актуальності магазину;

– дослідити ринок технічних товарів та інтернет-магазинів в Україні для визначення потреб клієнтів та особливостей конкурентів;

– вибір технології та засобів реалізації для створення проєкту;

– визначити функціональні вимоги до інтернет-магазину та його дизайну, з урахуванням результатів дослідження ринку та потреб клієнтів;

– написання технічного завдання;

– опис функціоналу;

– опис можливостей користувачів;

– створення інтерфейсу додатку;

– налаштування середовища розробки та технологій;

– написання коду для проєкту;

– провести тестування та відлагодження розробленого інтернет-магазину;

– розробка документації для програмного продукту.

Висновки до розділу 1

Було розглянуто аналоги існуючих інтернет-магазинів. Кожен з них має свої особливості, свої переваги та недоліки. Враховуючи сильні та слабкі сторони існуючих аналогів, можна покращити власну розробку, а саме: простий та

зрозумілий інтерфейс, швидке реагування на вирішення питань та проблем користувача, система лояльності користувачів, детальний і відповідний опис товару, захищеність приватної інформація та відгуки про товар.

РОЗДІЛ 2
СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення

Для проведення аналізу та визначення вимог до розроблюваного ПЗ необхідно:

– визначити предметну область розроблюваного програмного забезпечення і цілі які має досягнути ПЗ;

– потрібно зібрати всю необхідну інформацію про потенційних користувачів ПЗ, їх потреби та вимоги, а також про контекст, в якому ПЗ буде використовуватися;

– проаналізувати та розуміти потреби та вимоги користувачів, їх пріоритетність та взаємозв'язки між ними;

– визначити функціональні та нефункціональні вимоги до ПЗ;

– визначити технічні вимоги, тобто технічних характеристики системи, такі як апаратне та програмне забезпечення, архітектура, протоколи;

– створити сценарії та випадки використання, діаграми послідовності, прецедентів та інші необхідні UML моделі, які описують поведінку ПЗ та взаємодію з користувачами;

– зібрати всі отримані вимоги в єдиний документ, оформити їх у вигляді таблиць, діаграм та текстових описів.

Керуючись ціллю та предметною областю веб додатку, можна виявити такі вимоги:

– загальний опис: розроблена система є веб додатком для здійснення продажу технічних товарів. Програма повинна мати можливість відображати каталог товарів, детальний опис товару, відгуки користувачів, дані про стан замовлення, повинна мати можливість оформлювати замовлення та підтримувати користувачів, якщо виникатимуть проблеми із замовленням;

– інтерфейс користувача: система повинна мати інтуїтивно зрозумілий UI, простий у використанні та навігації. Інтерфейс має бути розроблений із сучасним і адаптивним макетом і повинен містити такі функції, як пошук, функція фільтрації товарів за категоріями;

– обробка платежів: система повинна містити функцію обробки платежів. Функціональність обробки платежів повинна бути інтегрована з публічним контролем доступу та повинна підтримувати декілька методів оплати;

– управління доступом: система повинна мати як публічний, так і приватний контроль доступу. Публічний доступ має бути доступним для всіх користувачів і мати можливість приймати оплату банківською карткою. Приватний доступ має бути доступний лише авторизованим користувачам і має включати додаткові функції, недоступні у відкритому доступі;

– керування користувачами: система повинна мати функцію керування користувачами для оперування обліковими записами користувачів, паролями та рівнями доступу. Функціональність керування користувачами повинна містити можливість додавати, редагувати та видаляти користувачів, а також призначати рівні доступу користувачам;

– адміністративна панель: система повинна мати можливість додавати та редагувати інформацію про товар, а також можливість керувати правами користувачів;

– безпека: система повинна включати функції безпеки для захисту даних користувача та запобігання несанкціонованому доступу. Функції безпеки повинні включати шифрування даних, автентифікацію користувача та контроль доступу;

– продуктивність: система повинна бути розроблена для роботи з великою кількістю користувачів, а також повинна мати можливість обробляти запити та оновлювати статуси в режимі реального часу;

– сумісність: система повинна бути сумісна з низкою веб-браузерів і пристроїв, включаючи настільні та мобільні пристрої. Система має бути розроблена таким чином, щоб бути швидкою та доступною, а також повинна

забезпечувати узгоджену взаємодію з користувачем на всіх пристроях і платформах.

Загалом розроблена система має бути надійною, масштабованою та зручною для користувача. Вона має надавати низку функцій і функціональних можливостей для продажу товарів, а також повинен легко обслуговувати велику кількість користувачів і їхні запити та замовлення.

2.1.1 Функціональні та нефункціональні вимоги

Також варто розглянути функціональні та нефункціональні вимоги.

Функціональні вимоги визначають, які функції або операції має виконувати система, програмний продукт або процес. Вони є ключовим елементом при розробці будь-якого продукту або системи, оскільки визначають його основні можливості та функції, які повинні бути реалізовані.

Функціональні вимоги:

– реєстрація користувачів. Додати можливість реєстрації нових користувачів і аутентифікації вже зареєстрованих користувачів;

– пошук товарів. Можливість пошуку товарів за назвою, категорією, описом і т.д;

– додавання товарів. Можливість додавання нових товарів до магазину;

– редагування товарів. Можливість редагувати інформацію про товари, додавати нові фотографії, змінювати ціну тощо;

– корзина. Можливість додавання товарів в кошик, перегляду та зміні кількості товарів в кошику перед оформленням замовлення;

– оформлення замовлення. Можливість оформити замовлення, вказати спосіб оплати та доставки, а також перевірити інформацію про замовлення перед його підтвердженням;

– оплата. Можливість здійснити оплату товарів, використовуючи різні способи оплати;

– відстеження замовлення. Можливість відстежувати статус замовлення, отримувати повідомлення про його обробку та доставку;

– підтримка клієнтів. Можливість звернутися до підтримки клієнтів для отримання допомоги, відповіді на запитання та вирішення проблем;

– адміністративна панель. Можливість управління магазином з адміністративної панелі, включаючи додавання/редагування/видалення товарів, відстеження замовлень, зміну статусу замовлень та зміну іншої інформації про магазин.

Нефункціональні вимоги є не менш важливим елементом вимог до розроблюваної програмної системи. Нефункціональні вимоги описують властивості, якості та характеристики, які має мати програмна система, щоб задовольнити потреби користувачів та відповідати стандартам та вимогам. Їх важливість полягає у тому, що вони визначають, як система повинна функціонувати з точки зору продуктивності, ефективності, безпеки, надійності, масштабованості та інших аспектів.

Нефункціональні вимоги:

– надійність: система повинна працювати стабільно, не зазнавати помилок і відмов. Це важливо для забезпечення безперебійної роботи магазину та збереження даних користувачів;

– швидкодія: інтернет-магазин повинен працювати швидко, швидко завантажувати сторінки, щоб користувачі не втрачали зацікавленість в товарах;

– безпека: магазин повинен забезпечувати безпеку від третіх осіб, які намагаються зламати або зловмисно отримати доступ до користувачів або даних магазину;

– наявність: магазин повинен бути доступний для користувачів 24 години на добу, 7 днів на тиждень;

– масштабованість: система повинна бути здатною масштабуватися, тобто збільшувати свої можливості і ресурси в залежності від потреб;

– наявність підтримки: магазин повинен мати добре організовану підтримку, яка може допомогти користувачам вирішувати проблеми та відповідати на питання;

– гнучкість: система повинна бути гнучкою в роботі з іншими інтернет-сервісами, включаючи можливість інтеграції зі сторонніми сервісами;

– легкість використання: інтерфейс магазину повинен бути зрозумілим і легко використовуваним користувачами.

Загалом, вимоги до системи спрямовані на створення зручного, безпечного та надійного веб-ресурсу для продажу товарів. Система має давати змогу користувачам обирати товари, оплачувати їх і відстежувати статус доставки. Крім того, користувачі повинні мати можливість обирати бажану мову інтерфейсу.

2.1.2 Моделі елементів програмного додатку у нотації UML

Моделі дозволяють графічно спроектувати логіку системи. Для побудови моделей необхідно визначити основні компоненти системи та їх взаємозв'язки, а також функції та операції, що виконуються кожним компонентом. Також необхідно визначити інтерфейси та структури даних, що використовуються для обміну інформацією між компонентами.

Першою розглянутою діаграмою – буде діаграма прецедентів розмежування логіки прав доступу до функціоналу сайту (рис. 2.1). Дана діаграма дає чітке поняття про те, як мають бути розмежовані ролі користувачів сайту. Вони мають бути добре продумані ще до початку етапу розробки та якісно реалізованими в коді додатку на час завершення. Для кожного виду зареєстрованого в системі користувачі буде свій інтерфейс та свій доступний функціонал. Так, наприклад, «користувачі» не матимуть доступу до функціоналу «адміністратора». А адміністратор в свою чергу має доступ до всіх сторінок сайту та матиме доступ до допоміжної сторінки «адміністративної-панелі». Ця діаграма не є вичерпною, але дає чітко усвідомити ролі користувачів системи.

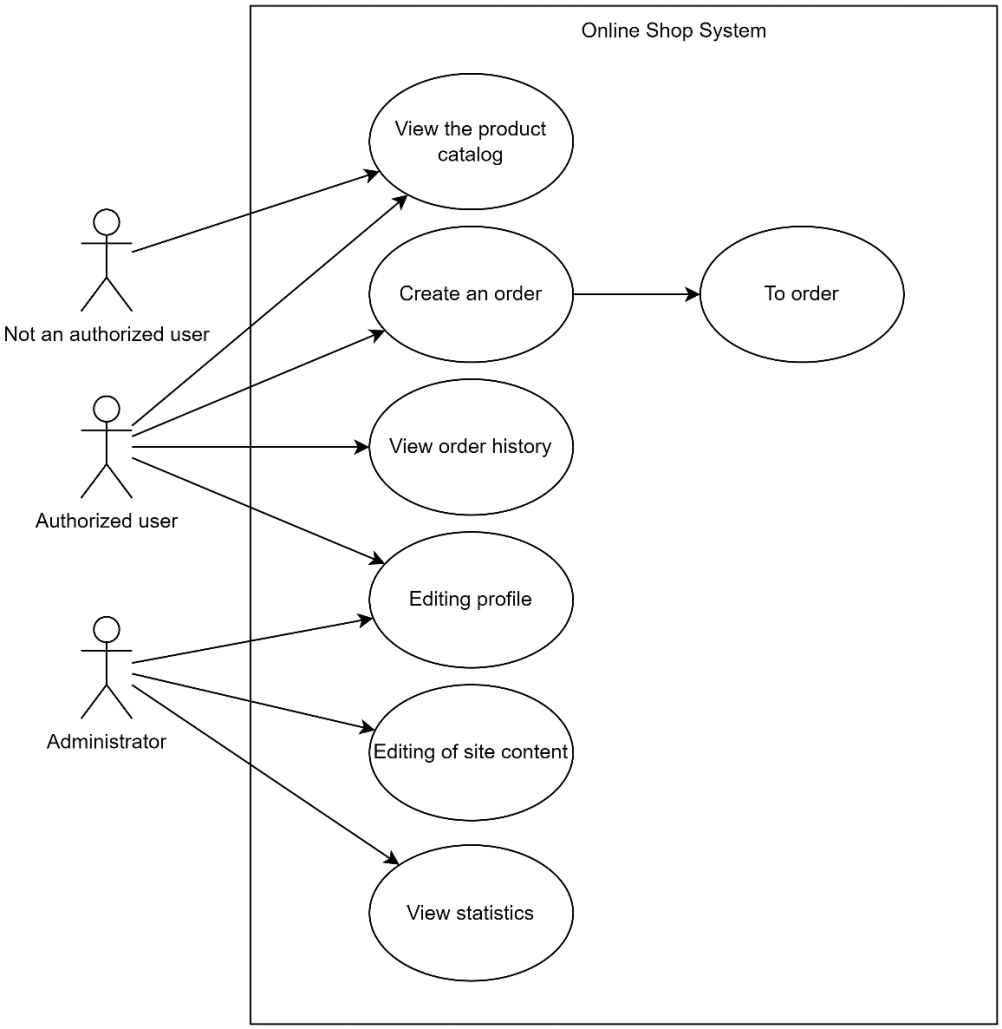

Рисунок 2.1 – Діаграма прецедентів різних ролей та прав доступу користувачів

На наступній представленій діаграмі (рис. 2.2) чітко представлено послідовність ітерацій, які виконує користувач в системі сайту від початку входження на нього до оформлення, підтвердження та оплати обраних товарів в замовлені.

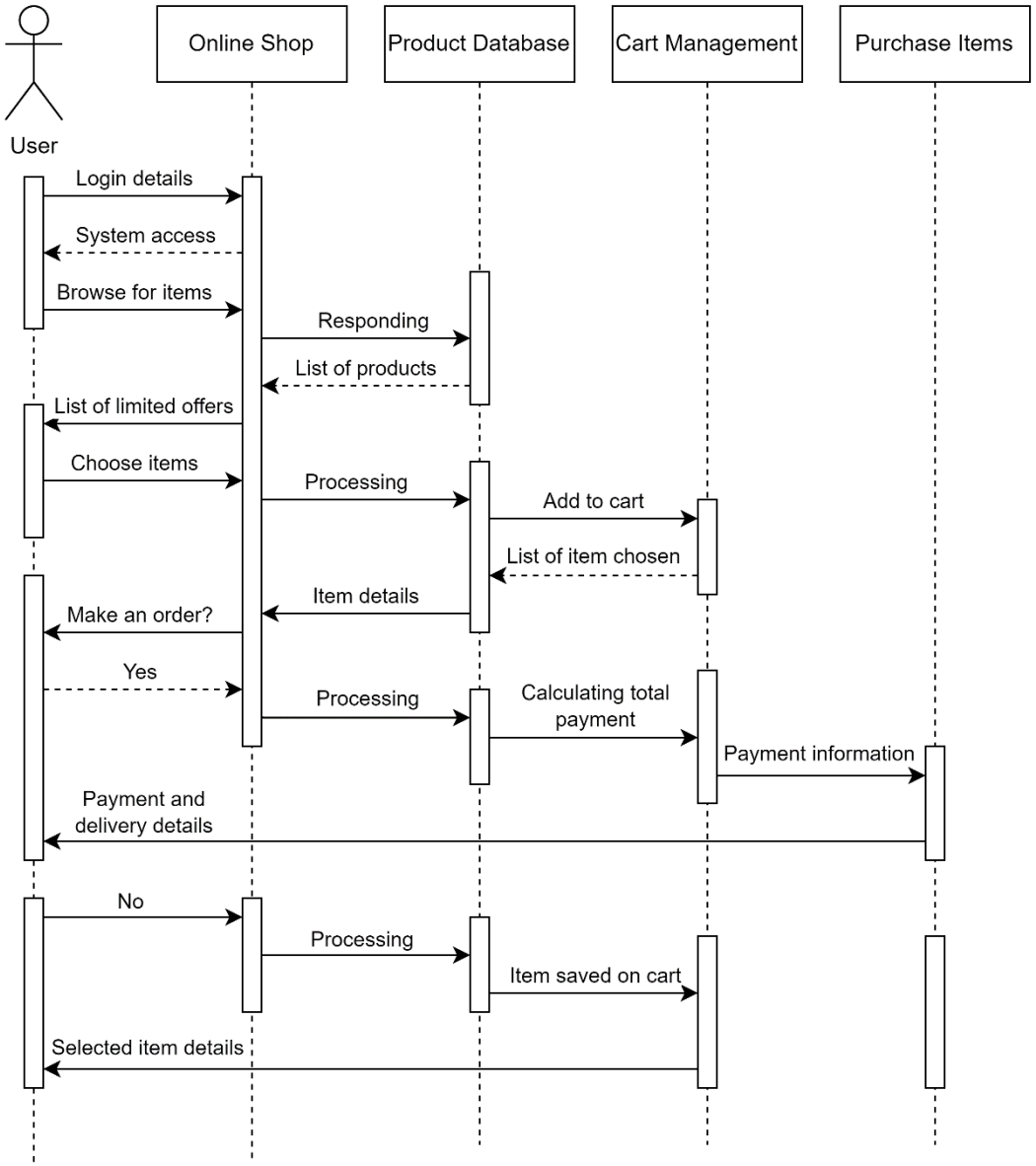

Рисунок 2.2 – Схема послідовності користування сайту до оформлення замовлення

2.1.3 Створення UX/UI-дизайну

Для створення дизайну були проаналізовані аналогічні веб магазини, та на основі них складені наступні схеми інтерфейсу. В будь якого інтернет магазину мають бути присутні чотири наступні сторінки, такі як: головна сторінка, сторінка з каталогом товарів, сторінка з описом конкретного товару, та сторінка оформлення замовлення.

Головна сторінка в проаналізованих магазинів об’єднує присутність меню з списком категорій, динамічна карусель з актуальними пропозиціями, та секцією з певними популярними товарами.

На рисунку 2.3 зображено макет головної сторінки сайту. В блоці «HEADER» буде розміщуватись логотип сайту, поле з пошуком товарів, меню з посиланнями на інші сторінки, посилання на корзину та відображення кількості товарів в ній та їхня вартість. Нижче буде розміщуватись список з категоріями товарів, поряд з ним динамічна карусель з акційними пропозиціями. Нижче розміщена секція із популярними товарами, та блок «FOOTER» в якому є різні важливі посилання на соц мережі та навігація по сайту.

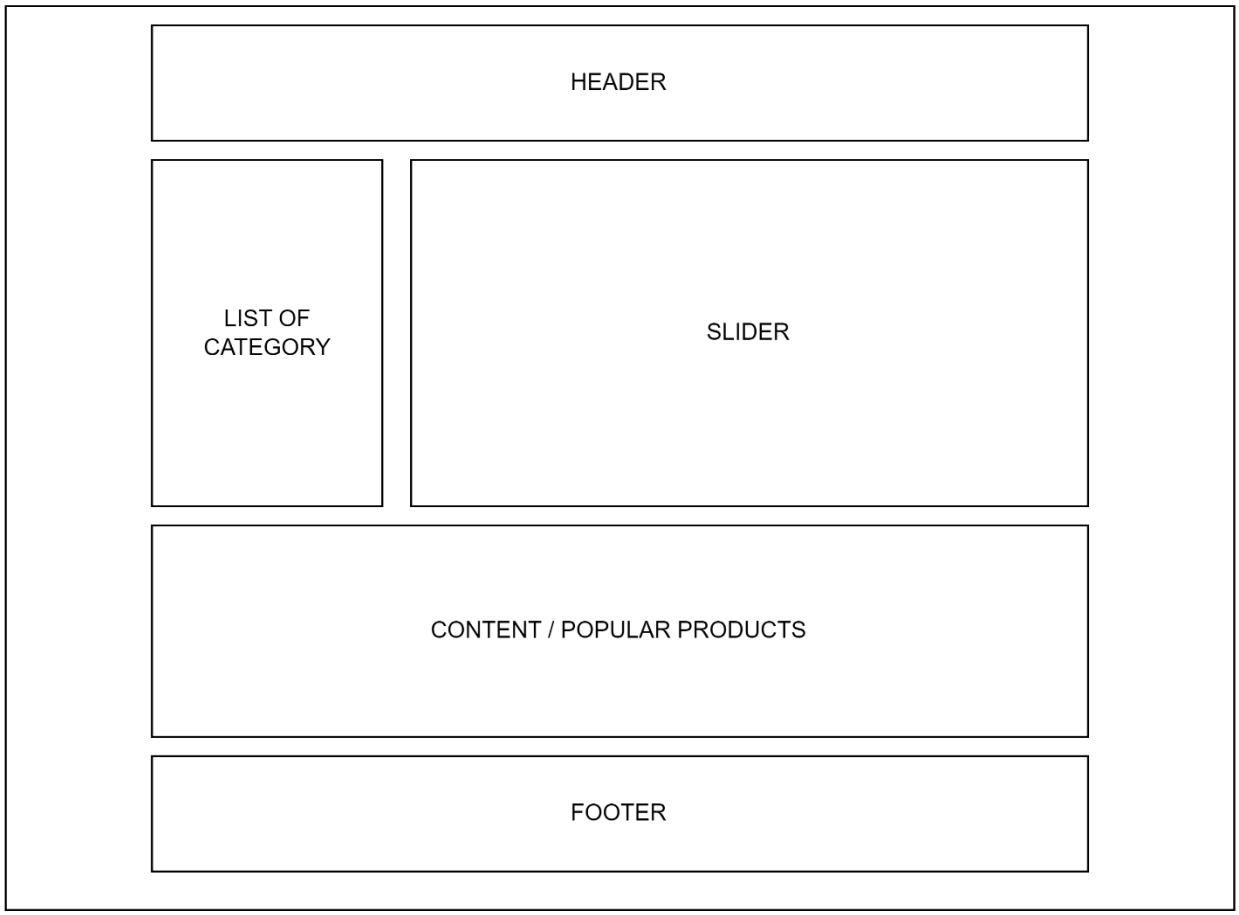

Рисунок 2.3 – Макет головної сторінки програми

На рисунку 2.4 зображено макет сторінки з каталогом товарів, де використовуючи різні можливості фільтрації підібрати необхідній товар.

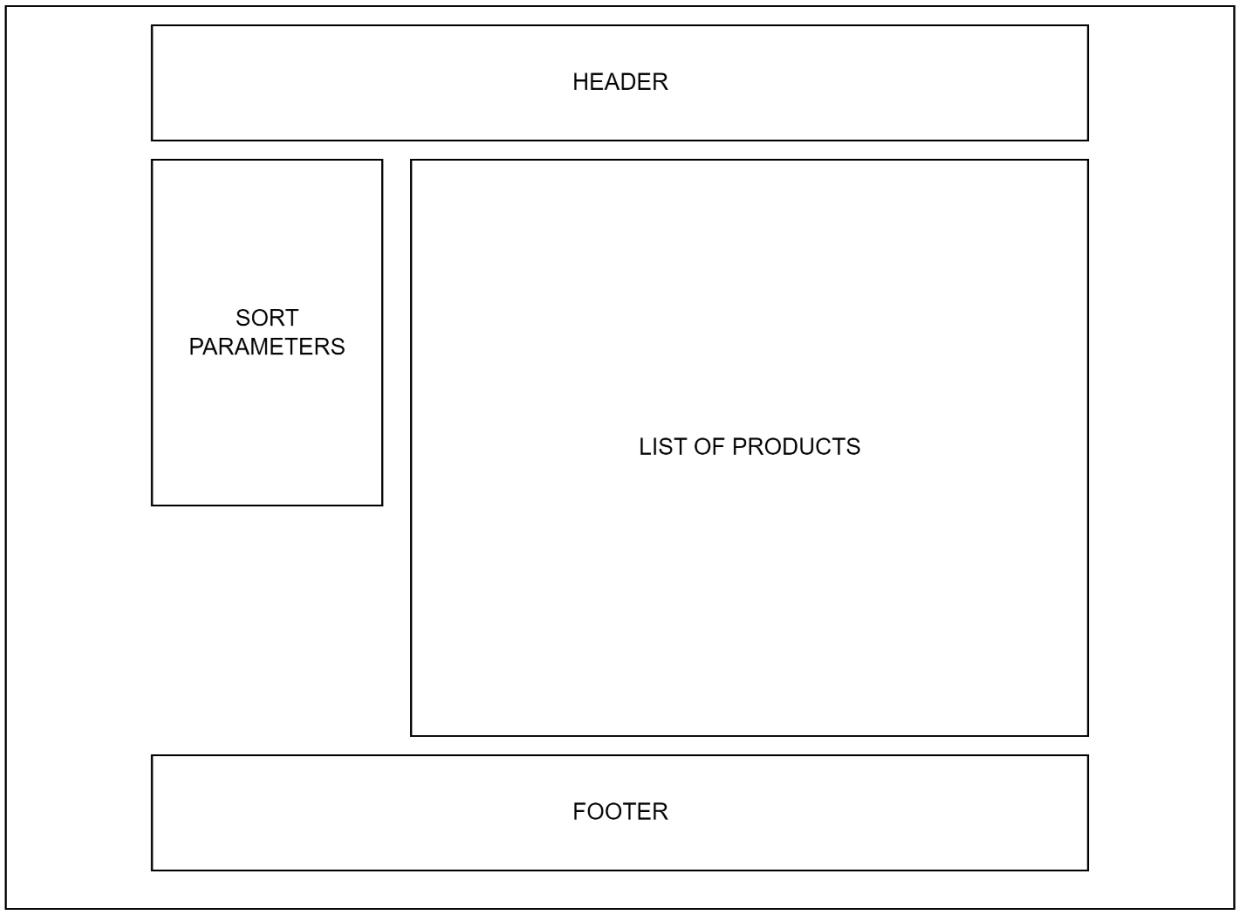

Рисунок 2.4 – Макет сторінки з каталогом товарів

На наступному рисунку 2.5 зображено макет сторінки з описом конкретного товару. На даній сторінці буде розміщуватись секція з детальними зображеннями товару, його описом та відгуками і оцінкою інших користувачів на даний товар. Також буде відразу вказати кількість одиниць даного товару до замовлення.

На сторінці із оформленням замовлення буде розміщуватись форми для заповнення даних про доставку та методи оплати, також можна буде переглянути замовлення перед фінальною оплатою (рис. 2.6).

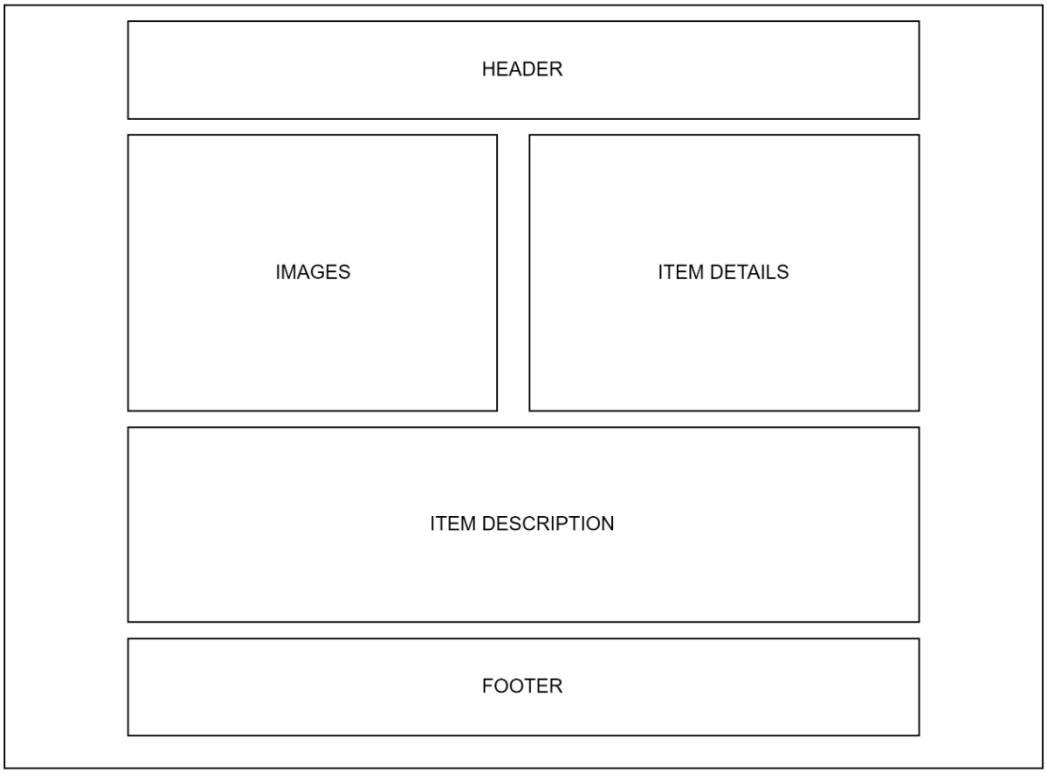

Рисунок 2.5 – Макет сторінки з описом конкретного товару

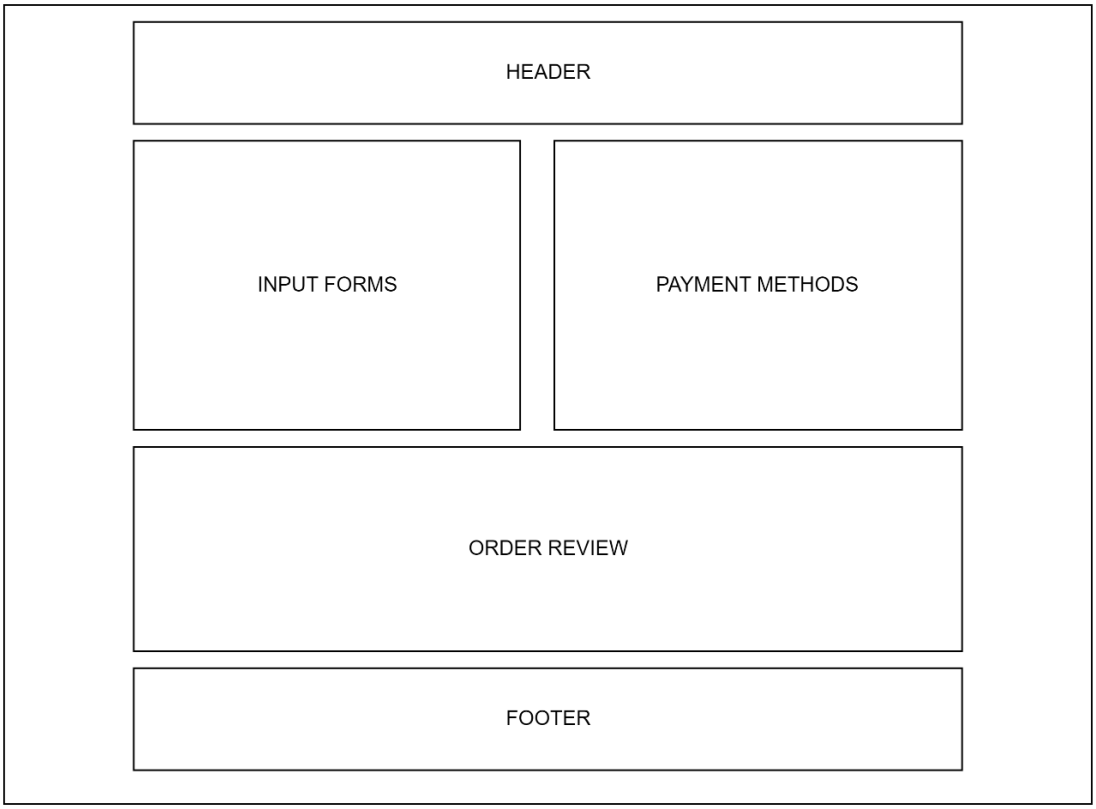

Рисунок 2.6 – Макет сторінки з оформленням замовлення

Сайт також містить другорядні сторінки такі як: часто запитувані питання, сторінка з описом магазину, сторінку з контактами, і також сторінки для адміністраторів для керуванням наповнення магазину.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Під час розробки програмної системи використовуються різноманітні методи і засоби, що дозволяють ефективно створювати, тестувати та підтримувати програмне забезпечення. Для обґрунтування вибору конкретних методів і засобів важливо враховувати багато факторів, зокрема характеристики проекту, вимоги до програмної системи, розмір команди розробників, доступність інфраструктури та багато іншого.

Наприклад, для створення програмної системи можуть бути використані різні мови програмування, такі як Java, Python, C++, або C#. Вибір конкретної мови програмування залежить від характеристик проекту, наприклад, типу програмної системи, складності розробки, вимог до швидкодії та надійності.

Також важливо вибрати потрібні засоби для версійного контролю, такі як Git або SVN, для забезпечення спільної роботи розробників та збереження історії змін. Для автоматизації тестування можуть використовуватися різні фреймворки, такі як Selenium, JUnit, або PyTest.

Крім того, для забезпечення ефективного управління проектом можуть бути використані різні засоби для спільної роботи, такі як Asana, Trello, або Jira.

Обґрунтування вибору конкретних методів і засобів повинно бути засноване на аналізі потреб проекту та розробників, а також на знаннях та досвіді команди розробників.

Перед вибором засобів та методів для розробки програмної системи слід враховувати наступні чинники:

– характеристики проекту: тип системи, що розробляється, його складність, розмір і потреби користувачів;

– кваліфікація команди розробників: знання мов програмування, досвід у розробці програмних проектів, здатність до використання нових засобів;

– вимоги до продукту: функціональні та нефункціональні вимоги до програмної системи, такі як швидкодія, надійність, безпека, масштабованість та інші;

– інфраструктура: доступність інфраструктури для розробки, тестування та виконання програмної системи;

– масштаб проекту: розмір команди розробників, обсяг роботи, часові рамки та інші параметри, що впливають на процес розробки програмної системи;

– вартість: вартість розробки, тестування та підтримки програмної системи, а також вартість засобів та інфраструктури, що використовуються;

– взаємодія з користувачем. Додаток має бути розроблено таким чином, щоб забезпечувати плавну та інтуїтивно зрозумілу взаємодію з користувачем, із зрозумілими та простими у використанні інтерфейсами та корисними повідомленнями зворотного зв’язку;

– сумісність та інтеграція: можливість інтеграції програмної системи з іншими системами, сумісність з різними операційними системами та іншими засобами.

У розробці програмної системи для кваліфікаційної роботи бакалавра було обрано ряд методів та засобів для кращої та зручної розробки.

Одним із основних засобів була використана мова Python. Це високорівнева мова програмування, яка має велику кількість бібліотек і фреймворків для розробки веб-додатків [1].

Основні особливості Python включають:

– інтерпретованість: програми Python виконуються на підтримуваному інтерпретаторі, що дозволяє швидко писати, тестувати та відлагоджувати код;

– об'єктно-орієнтована парадигма: Python підтримує об'єктно-орієнтоване програмування, що дає можливість організувати код у вигляді класів та об'єктів;

– широкий спектр бібліотек та фреймворків: Python має велику кількість бібліотек та фреймворків, що спрощують розробку різноманітних застосувань, таких як наука даних, машинне навчання, веб-розробка, автоматизація та багато іншого;

– простота та зрозумілість: синтаксис Python дуже простий та лаконічний, що дозволяє легко зрозуміти код, навіть новачкам у програмуванні;

Для розробки інтернет-магазину був обраний фреймворк Django [2], [6], [7].

Django – це високорівневий веб-фреймворк, написаний на мові програмування Python, який дозволяє швидко та ефективно створювати складні веб-додатки. Основні особливості Django:

– вбудована система адміністрування, що дозволяє легко створювати та змінювати моделі даних, а також взаємодіяти з ними через веб-інтерфейс [17];

– вбудована ORM (об'єктно-реляційна маппінг), яка дозволяє працювати з базою даних через Python-об'єкти замість написання SQL-запитів вручну [13];

– можливість створення веб-додатків з різним рівнем складності, від невеликих сайтів до великих корпоративних систем;

– велика кількість розширень та модулів, які дозволяють розширювати можливості фреймворку та прискорювати процес розробки [16];

– Django використовує шаблонізатор, який дозволяє відділяти логіку від представлення та створювати динамічні HTML-сторінки [9];

– Django має вбудовану підтримку безпеки, включаючи захист від кросс-сайтових атак та інших видів атак на веб-додаток.

В основі фреймворка лежить шаблон MVC (рис. 2.7). MVC (модель-вигляд-контролер) – архітектурний шаблон, що використовується під час проєктування та безпосередньо розробки програмного забезпечення. Цей шаблон є доволі популярним тому що він поділяє всю систему на три частини: модель даних, вигляд даних і керування даними. Завдяки цьому зміна одного з цих компонентів мінімально впливає на інші або зовсім не впливає. Модель (Model) призначена для зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний

за представлення даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача, і відправляє відповідь користувачу у вигляді сторінки з даними (View). Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами [4].

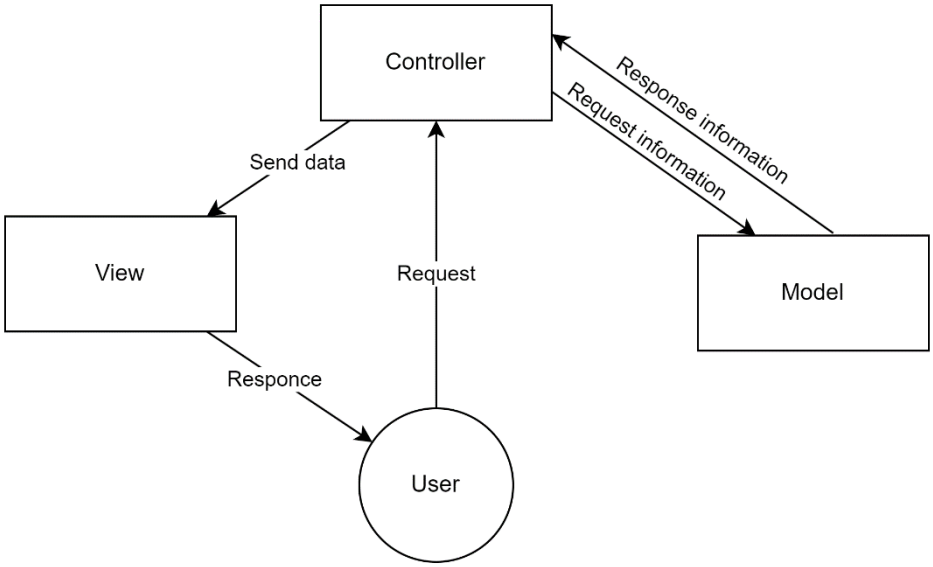

Рисунок 2.7 – Принцип роботи MVC шаблона

Узагальнюючи, Django є потужним та популярним веб-фреймворком, який дозволяє розробникам ефективно та швидко створювати веб-додатки з високою функціональністю та безпекою.

База даних – для зберігання даних про товари, замовлення та інші аспекти магазину можна використовувати PostgreSQL. PostgreSQL (часто зазначається як "Postgres") є вільною та відкритою реляційною системою управління базами даних (СУБД). Він підтримує широкий спектр функцій, включаючи транзакції, зовнішні ключі, підтримку відомих стандартів SQL, реплікацію та багатокористувацький доступ [3], [10], [13].

PostgreSQL є однією з найбільш потужних та розширюваних СУБД, що дозволяє використовувати її для обробки великих обсягів даних і високонавантажених додатків. Вона підтримує багатопоточну обробку запитів, оптимізовану роботу з індексами та можливість підключення додаткових модулів.

PostgreSQL має широке співтовариство користувачів та розробників, яке допомагає забезпечувати постійну підтримку та розвиток системи. Його можна використовувати на різних операційних системах, включаючи Windows, Linux та Mac OS. Також PostgreSQL є базою даних за замовчуванням для деяких відомих програм, таких як OpenStreetMap та Skype.

Для динамічності та плавності роботи сторнок було використано мову програмування JavaScript. JavaScript є мовою програмування, яка зазвичай використовується для створення динамічних веб-сторінок та додавання інтерактивності до веб-сайтів. Вона була створена в 1995 році компанією Netscape і з тих пір стала однією з найпопулярніших мов програмування у світі. JavaScript підтримується всіма сучасними браузерами, що дозволяє розробникам створювати різноманітні програми та додатки для веб-середовища [8], [14], [11].

Основні характеристики JavaScript:

– JavaScript є мовою програмування, що працює на клієнтській стороні (в браузері) і на серверній стороні (за допомогою платформи Node.js);

– вона є мовою програмування з високим рівнем абстракції, що дозволяє розробникам писати складні програми з мінімальними зусиллями;

– JavaScript використовується для створення динамічних ефектів на веб-сторінках, таких як анімація, зміна кольору, розміру та положення елементів сторінки;

– вона підтримує об'єктно-орієнтований, функціональний та імперативний стиль програмування;

– JavaScript має широкий набір бібліотек та фреймворків, таких як jQuery, React і Angular, що допомагають розробникам швидше розробляти веб-додатки.

Загалом, JavaScript є надзвичайно корисною мовою програмування для веб-розробки, що дозволяє розробникам створювати більш інтерактивні, цікаві та корисні веб-додатки.

Також була використана мова текстової розмітки HTML та мова стилів CSS. HTML (HyperText Markup Language) – це мова розмітки, що використовується для

створення структури веб-сторінки. За допомогою HTML можна створювати різні елементи на сторінці, такі як заголовки, параграфи, списки, таблиці, зображення, посилання та інші. HTML визначає зміст сторінки та розміщення її елементів.

CSS (Cascading Style Sheets) – це мова стилів, що використовується для задання зовнішнього вигляду веб-сторінки. За допомогою CSS можна змінювати кольори, шрифти, розміри та розташування елементів на сторінці. CSS дозволяє створювати привабливий та привабливий вигляд сторінки, що забезпечує більш зручне та естетичне сприйняття інформації користувачами.

Таким чином, HTML використовується для створення змісту веб-сторінки, тоді як CSS використовується для задання її вигляду та стилю. Обидві мови є важливими для розробки веб-сайтів та їх оформлення, і вони використовуються разом, щоб створити функціональний та привабливий дизайн сторінки.

Висновки до розділу 2

Сформульовано та визначено вимоги до розробленої системи. Розроблена система являє собою веб-ресурс, створений з використанням технологій Python, Django, PostgreSQL та JavaScript. Метою даного веб-ресурсу є надання всім користувачам зручної та ефективної платформи для купівлі товарів.

РОЗДІЛ 3
РОЗРОБКА ВЕБ-САЙТУ ДЛЯ ПРОДАЖУ ТЕХНІЧНИХ ТОВАРІВ З ВИКОРИСТАННЯМ DJANGO

3.1 Практична реалізація об’єкта проектування

VS Code (Visual Studio Code) – це безкоштовне середовище розробки, розроблене компанією Microsoft. Воно стало одним з найпопулярніших інструментів серед розробників завдяки своїй простоті, широкому спектру можливостей та активній спільноті користувачів.

Visual Studio Code має ряд важливих наступних переваг які виділяють цей редактор коду серед інших:

– мультиплатформенність. VS Code доступний на Windows, macOS та Linux, що дозволяє розробляти програми на різних операційних системах;

– підтримка багатьох мов програмування. VS Code надає підтримку для широкого спектру мов програмування, включаючи C++, C#, Java, Python, JavaScript, PHP, Ruby та багато інших;

– розширюваність. VS Code має потужну систему розширень, яка дозволяє розробникам налаштовувати середовище під свої потреби. Існує велика кількість розширень, які можна встановити з веб-магазину VS Code;

– інтеграція з системами контролю версій. VS Code легко інтегрується з різними системами контролю версій, такими як Git, і надає розширені можливості роботи з репозиторіями;

– вбудовані інструменти для розробки. VS Code має багато корисних функцій, таких як розумний автозаповнювач, вбудована консоль, зручний дебагер, можливість швидкого перегляду документації та багато інших, що полегшують розробку програм;

– відладка. VS Code підтримує потужні засоби відладки, що дозволяють розробникам відслідковувати та виправляти помилки в коді;

– швидкість та продуктивність. VS Code працює швидко навіть з великими проектами, завдяки своїй оптимізованій архітектурі.

VS Code є чудовим вибором для розробки інтернет-магазину на мові Python з використанням фреймворку Django. Ось деякі переваги в можливостях які надає VS Code, специфічні для цього типу розробки:

– розширення для Django. VS Code має розширення, спеціально призначені для розробки Django-проектів. Наприклад, розширення Python та Django дозволяють вам отримати доступ до функціональності Django, такої як підказки коду, автозаповнення шаблонів, підтримка мови шаблонів Django та багато іншого;

– інтегрований термінал. VS Code має вбудований термінал, що дозволяє виконувати команди Django-утиліт (наприклад, manage.py) безпосередньо з середовища розробки. Це полегшує виконання міграцій бази даних, запуск сервера розробки Django та інші процеси;

– відлагодження (Debugging). VS Code має потужну систему відлагодження, яка підтримує розробку Django-проектів. Ви можете встановлювати точки зупинки (breakpoints) у вашому коді та стежити за його виконанням, а також аналізувати значення змінних у режимі відлагодження;

– Git-інтеграція. VS Code має вбудовану підтримку системи контролю версій Git, що дозволяє легко відстежувати зміни в вашому проекті. Ви можете переглядати, змінювати та комітити код безпосередньо з інтерфейсу VS Code;

– автозаповнення та підказки коду. VS Code надає розумний автозаповнювач та підказки коду для мови Python та Django. Ви отримуєте швидкий доступ до методів та атрибутів Django-об'єктів, що полегшує роботу з фреймворком;

– вбудована підтримка віртуальних середовищ (Virtual Environments). VS Code дозволяє легко налаштувати та використовувати віртуальні середовища для вашого проекту Django. Це сприяє відокремленості залежностей проекту та забезпечує чистоту середовища розробки;

– розширюваність. VS Code має широкий вибір розширень, які полегшують розробку Django-проектів. Наприклад, розширення для форматування коду, перевірки якості, роботи з базами даних та багато інших [5].

Для створення проекту необхідно завантажити з офіційного сайту одну з підтримуваних версій Python [1], [12]. Після встановлення в терміналі комп'ютера можна викликати команду «`python -version`» для перевірки коректного встановлення усіх залежностей (рис. 3.1).

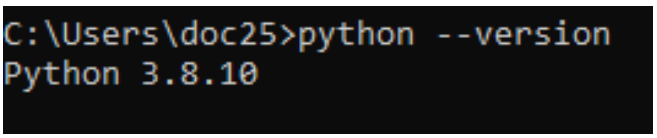

Рисунок 3.1 – Перевірка коректного встановлення Python

Наступним кроком буде створення віртуального середовища. Віртуальні середовища Python є корисним інструментом для ізоляції та керування залежностями проектів Python. Ось кілька причин, для чого вони потрібні:

– розділення проектів. Віртуальне середовище дозволяє вам створити ізольоване середовище для кожного вашого проекту. Це означає, що кожен проект може мати свої власні версії пакетів та залежностей, не конфліктувати між собою;

– управління залежностями. Ви можете легко встановлювати, оновлювати та видаляти пакети, що використовуються в вашому проекті, в межах віртуального середовища. Це дозволяє зберігати чистоту та консистентність залежностей вашого проекту;

– відокремлення версій Python. Ви можете використовувати різні версії Python для різних проектів за допомогою віртуальних середовищ. Це особливо корисно, коли вам потрібно розробляти проекти, що вимагають різних версій Python або коли ви переходите на новішу версію Python і бажаєте перевірити сумісність вашого коду;

– портативність. Віртуальні середовища дозволяють вам розповсюджувати ваш проект разом з його залежностями. Це означає, що інші розробники можуть

легко встановити та запустити ваш проект на своїх системах, не переживаючи проблем зі сумісністю залежностей;

– тестування. Віртуальні середовища дозволяють вам легко налаштовувати окреме середовище для тестування вашого коду. Ви можете ізолювати ваші тести від основного середовища розробки, забезпечуючи чистоту тестового середовища та уникнення можливих впливів на інші частини вашого проекту.

Для створення віртуального середовища потрібно виконати наступну команду «`python -m venv *назва_віртуального_середовища*`».

Після чого потрібно активувати віртуальне середовище наступною командою «`.\*назва_віртуального_середовища*\Scripts\activate`».

Наступним кроком буде встановлення фреймворку Django. Для цього відкриємо командний рядок або термінал та виконаємо команду «`pip install django`».

Далі потрібно відкрити термінал в каталозі, де бажаємо створимо проект. Потім виконаємо команду «`django-admin startproject *назва_проекту*`». Ця команда створить каталог з вказаним іменем проекту та необхідними файли Django.

Запустимо сервер розробки Django: У командному рядку або терміналі потрібно перейти до каталогу створеного проекту. Виконаймо команду «`python manage.py runserver`», щоб запустити сервер розробки Django. Після цього можна переглядати проект, відкривши веб-браузер і перейшовши за адресою http://localhost:8000.

На даному етапі можна приступати до розробки інтернет-магазину, створюючи моделі, представлення, шаблони та додавання логіки, необхідної для проекту. Щоб отримати детальніше розуміння фреймворку та його можливостей, можемо ознайомитися з офіційною документацією Django.

По завершенню розробки структура проекту виглядає наступним чином (рис. 3.2).

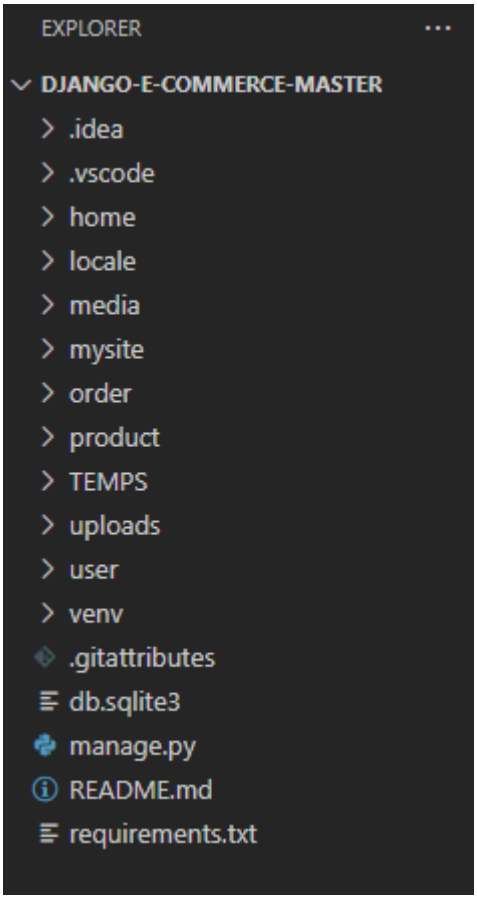

Рисунок 3.2 – Структура проекту в середовищі розробки VS Code

Коренева папка проекту. Це головна папка проекту, яка містить всі компоненти та налаштування проекту. В даному випадку це папка «mysite», у цій папці знаходяться файли manage.py (використовується для керування проектом) та settings.py (налаштування проекту) [15], [16].

Django організований на основі додатків. У директорії проекту можуть бути додатки, які створюються для своїх потреб. Кожен додаток містить свою власну директорію з іменем додатку, в якій зазвичай містяться файли моделей, представлень, URL-ів, шаблонів та інших компонентів, пов'язаних з цим додатком. Для створення нового додатку використовується команда «`python manage.py startapp *назва_додатку*`».

Файл models.py в директорії кожного додатку містить опис моделей бази даних, які використовуються в проекті. Моделі Django описують сутності додатку,

такі як користувачі, товари, замовлення тощо, та визначають їх поля та взаємозв'язки.

Файл views.py містить представлення (views) – функції або класи, які обробляють HTTP-запити та відповідають на них. Представлення обробляють логіку бізнес-процесів, взаємодіють з моделями, формують дані для відображення та повертають відповіді клієнту.

У директорії кожного додатку можуть знаходитися директорії з шаблонами (templates), де зберігаються файли HTML для відображення даних. Шаблони дозволяють розділити логіку відображення та бізнес-логіки, дозволяючи створювати динамічні та настроюванні сторінки.

Файл urls.py в директорії проекту та в директорії кожного додатку містить маршрутизацію URL-ів – зв'язок між URL-шляхами та відповідними представленнями. В цих файлах визначається, яке представлення повинно бути викликане при обробці певного URL-запиту.

На головній сторінці сайту розміщений список з категоріями, також розміщений слайдер з певними пропозиціями та знижками на товари, популярні товари та блок з підпіркою схожих товарів які підходять користувачу (рис. 3.3).

В шапці сторінки розміщена навігація по інших сторінках сайту, можливість зміни мови та валюти. Також логотип, поле для пошуку, інформація про аккаунт користувача та картка користувача.

В підвальній частині сторінки для зручності продубльована навігація для зручності та можливість підписатися на останні пропозиції магазину.

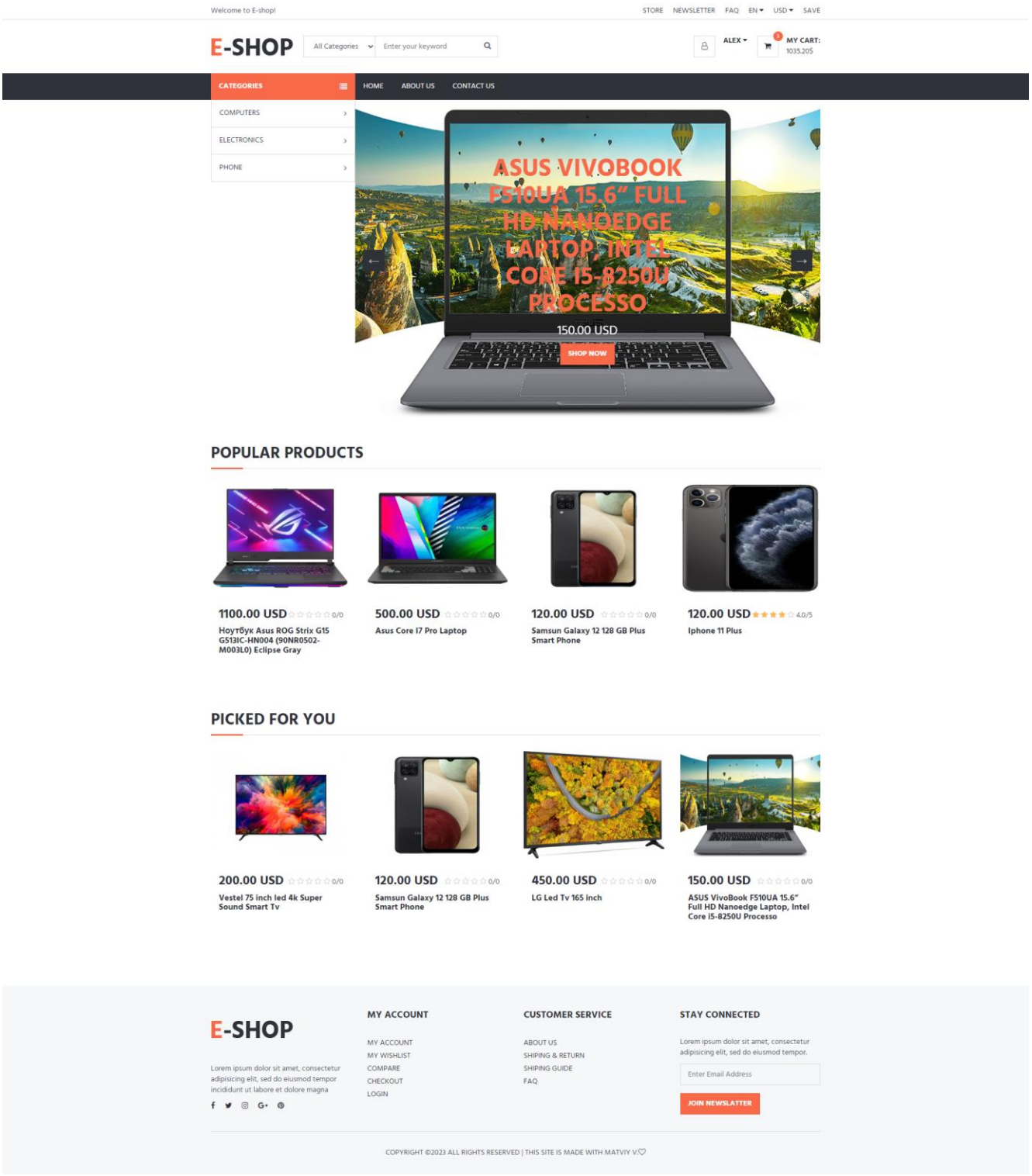

Рисунок 3.3 – Головна сторінка сайту

При необхідності користувач може перейти за посиланням на конкретний товар для перегляду детальнішого опису товару та можливістю додавання його в кошик, після чого переходити до оформлювати замовлення (рис. 3.4). Також можна

детальніше роздивитися товар, адже присутні більше фото та можливість збільшення фото.

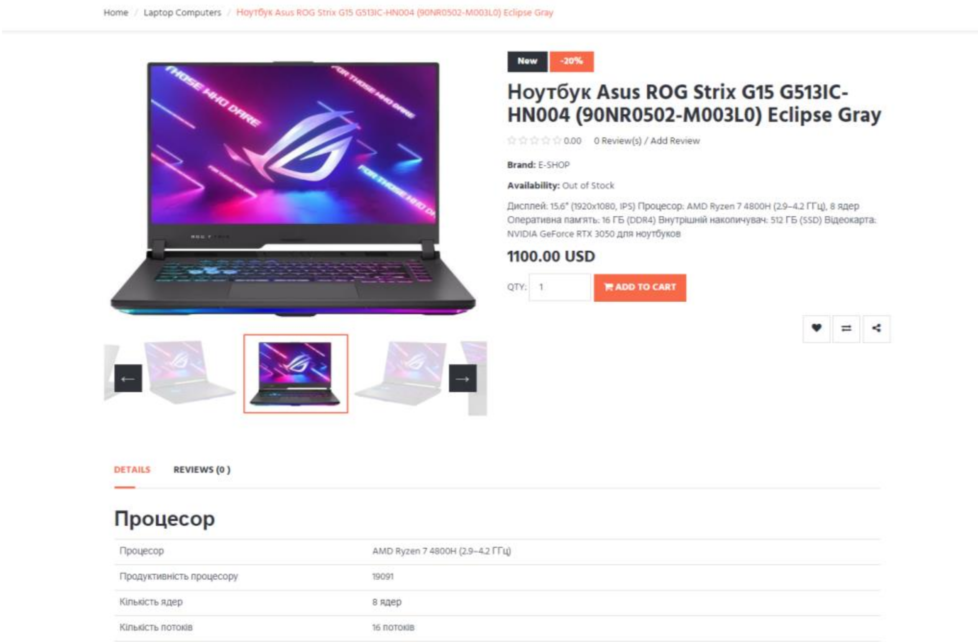

Рисунок 3.4 – Сторінка з детальним описом товару

До кожного товару користувач може написати свій відгук, що допоможе зрозуміти іншим користувачам як даний товар показує себе у практичному використанні (рис. 3.5).

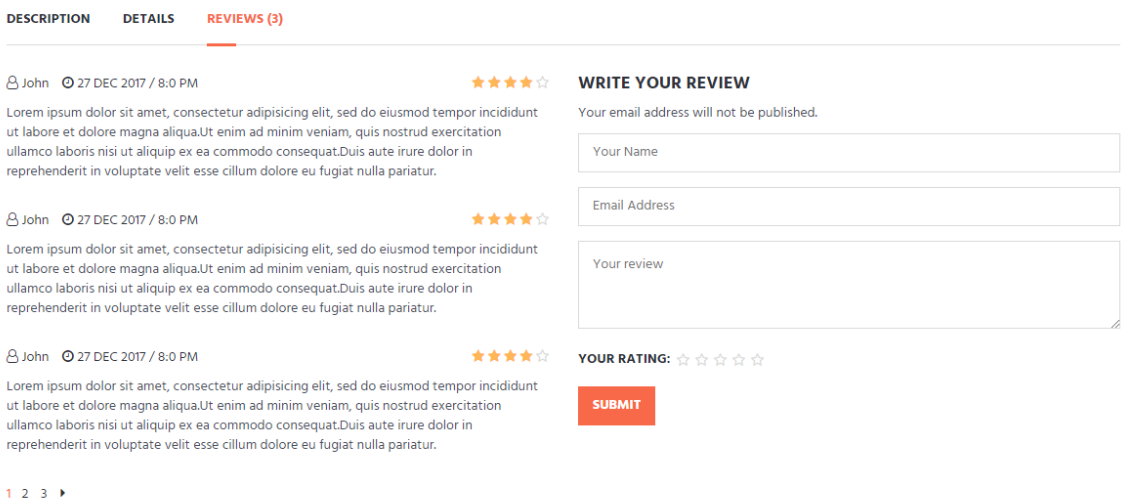

Рисунок 3.5 – Відгуки користувачів про товар

Однією з не менш важливих сторінок є сторінка з переглядом каталогу товарів, адже з нею користувачі будуть часто взаємодіяти. На даній сторінці розміщена певна фільтрація товарів, що дозволяє швидше знаходити необхідний користувачу товар (рис. 3.6).

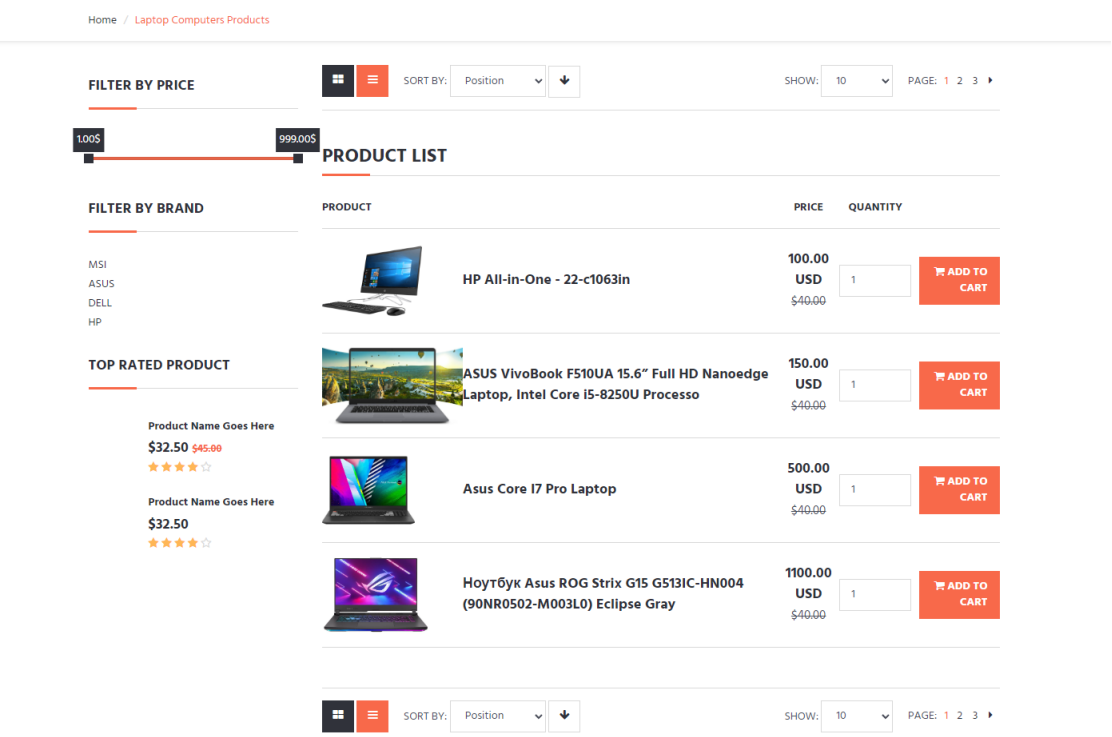

Рисунок 3.6 – Каталог з товарами

Після того як користувач додав необхідні товари собі у кошик він може перейти до кошика, подивитися загальну вартість покупки та перейти до оформлення замовлення (рис. 3.7). При оформлені замовлення користувач має надати необхідну інформацію, таку як місце доставки та спосіб доставки (рис. 3.8).

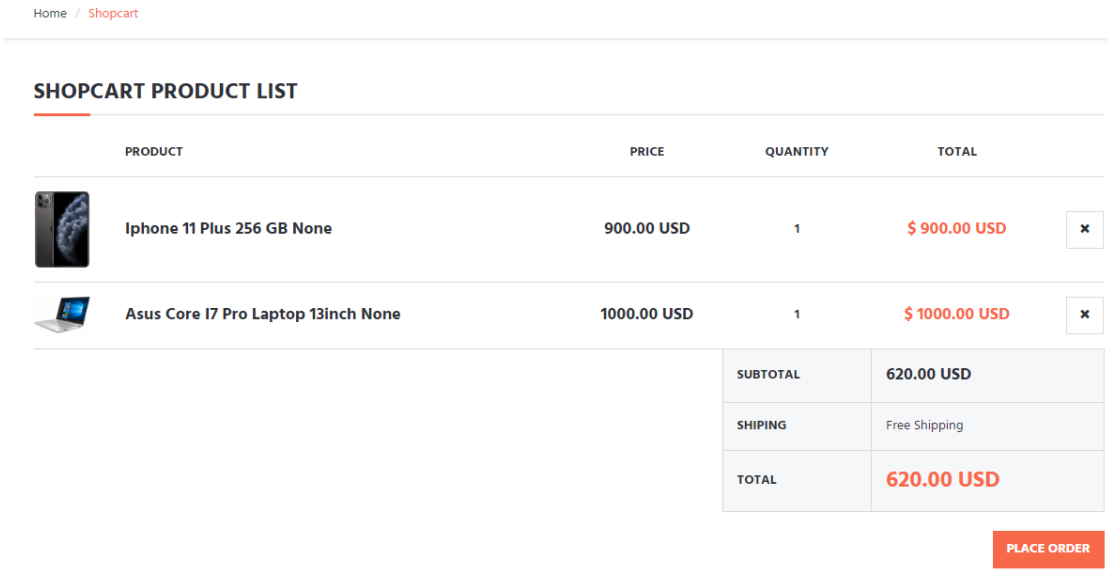

Рисунок 3.7 – Картка користувача

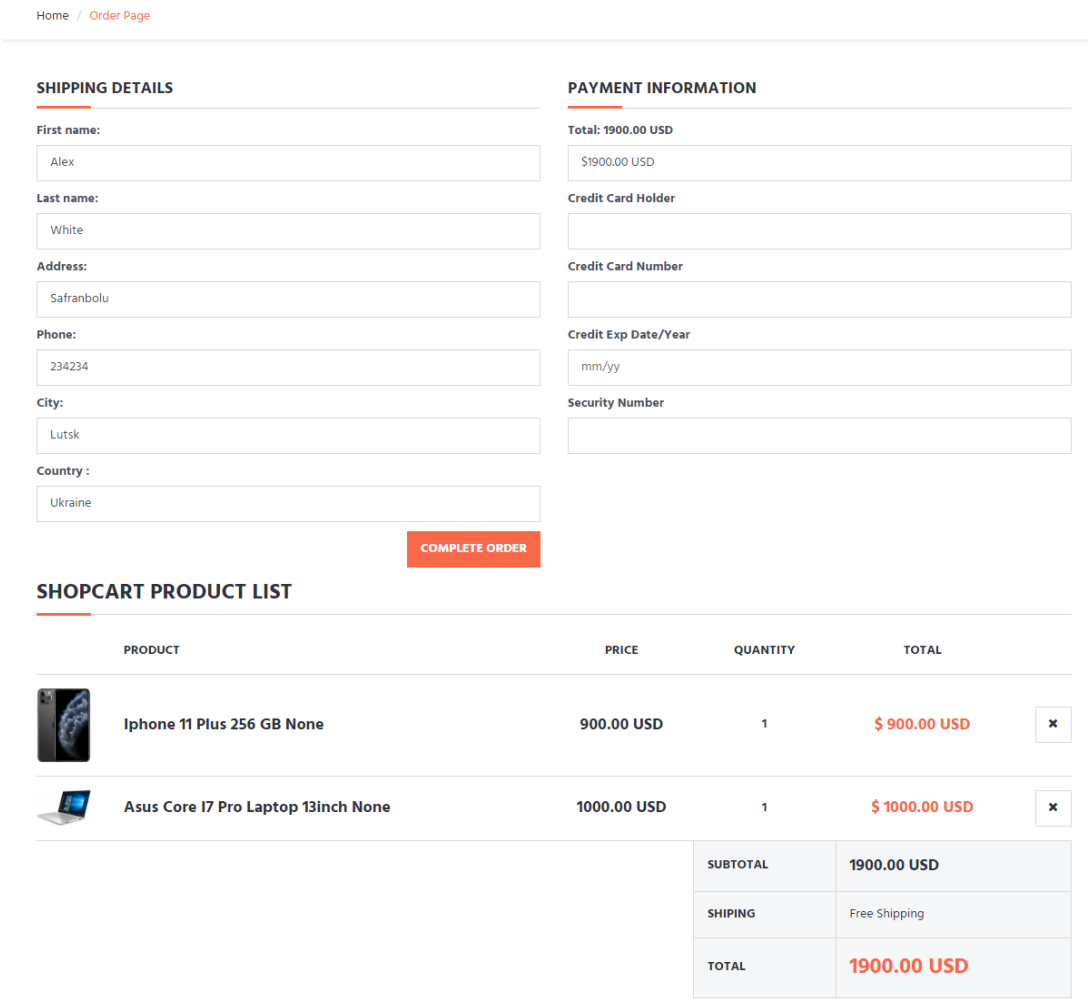

Рисунок 3.8 – Оформлення замовлення

Після успішного заповнення усієї інформації користувач отримає повідомлення про успішно завершену покупку (рис. 3.9).

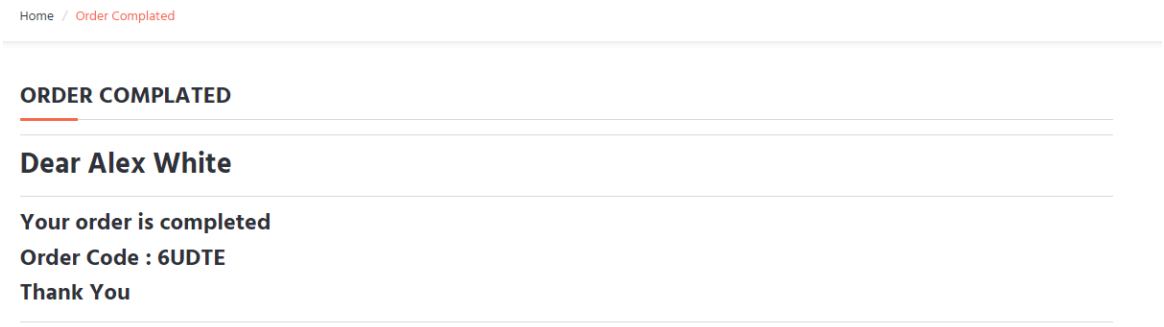

Рисунок 3.9 – Успішне оформлене замовлення

Також користувач має можливість перейти до свого профілю та переглянути свою особисту інформацію та можливість її змінити (рис. 3.10). Також він може переглянути список своїх замовлень (рис. 3.11).

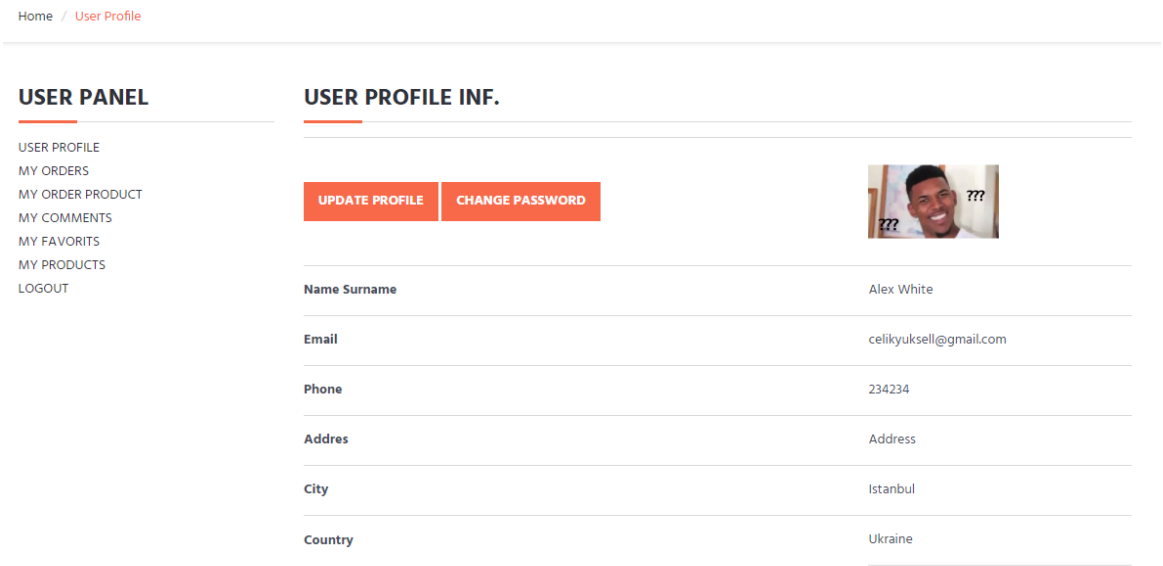

Рисунок 3.10 – Особистий кабінет користувача

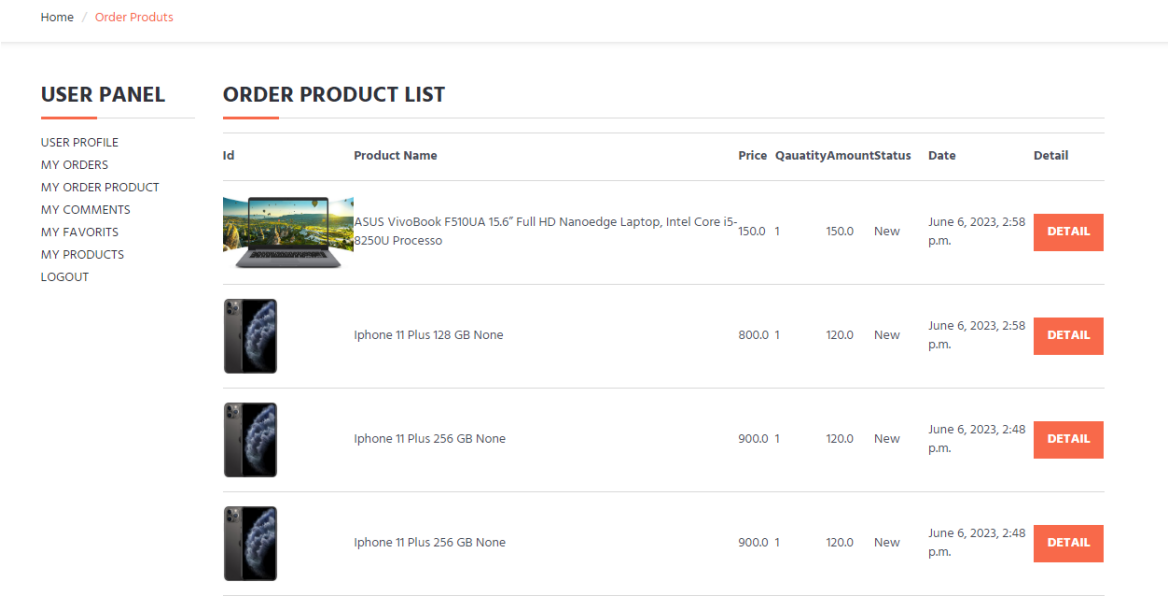

Рисунок 3.11 – Список замовлень користувача

На сайті також присутні і інші допоміжні сторінки як сторінка з описом про магазин, сторінка з контактною формою та сторінка з часто запитуваними питаннями.

3.2 Розробка бази даних

Для розробки бази даних була обрана технологія баз даних PostgreSQL. База даних PostgreSQL є однією з найпопулярніших і потужних відкритих реляційних баз даних. Ось кілька переваг, які роблять PostgreSQL привабливим вибором для багатьох проектів:

– надійність і стабільність. PostgreSQL відомий своєю надійністю і стабільністю. Він використовує транзакційну модель забезпечення цілісності даних ACID (Atomicity, Consistency, Isolation, Durability) і має вбудовану підтримку резервного копіювання та відновлення, що забезпечує безпеку даних;

– розширюваність. PostgreSQL підтримує горизонтальне та вертикальне масштабування, що дозволяє розширювати базу даних, додавати нові вузли і збільшувати продуктивність. Він також має можливості розподіленої обробки даних та реплікації;

- розширені можливості. PostgreSQL має багатий набір функцій і можливостей. Він підтримує різноманітні типи даних, включаючи географічні дані, JSON, XML та багато інших. Також він має велику кількість вбудованих функцій і розширень, які дозволяють розширити його функціональність;

– підтримка мов програмування. PostgreSQL підтримує багато мов програмування, включаючи SQL, PL/pgSQL, Python, Java, C/C++, Ruby та інші. Це дає розробникам більше гнучкості при роботі з базою даних і інтеграції з іншими компонентами системи;

– спільнота та підтримка. PostgreSQL має велику активну спільноту користувачів і розробників, яка забезпечує підтримку, допомогу та розвиток бази даних. Ви можете знайти багато документації, форумів, блогів та ресурсів, щоб отримати допомогу та поради;

– відкритий код і безкоштовність. PostgreSQL є відкритим програмним забезпеченням, що означає, що ви можете користуватися ним безкоштовно і мати можливість переглядати і змінювати його вихідний код. Це також дозволяє широку спільноту розробників співпрацювати та вносити свої внески у вдосконалення бази даних.

Загалом, PostgreSQL є потужною і надійною реляційною базою даних з розширеними можливостями і підтримкою спільноти. Вона є привабливим вибором для проектів будь-якого розміру і складності.

Для користуванням даною базою даних був завантажений та інстальований Docker. За допомогою даного програмного рішення можна швидко та легко налаштувати підключення до віртуального середовище з ядром бази даних.

Контейнери Docker забезпечують ізольоване середовище для PostgreSQL, що допомагає уникнути конфліктів з іншими додатками або системними ресурсами. За З Docker можна легко управляти доступом до контейнера та налаштовувати мережеві правила для забезпечення безпеки бази даних.

База даних проекту налічує двадцять пов’язаних між собою таблиць та дві незалежних (рис. 3.12).

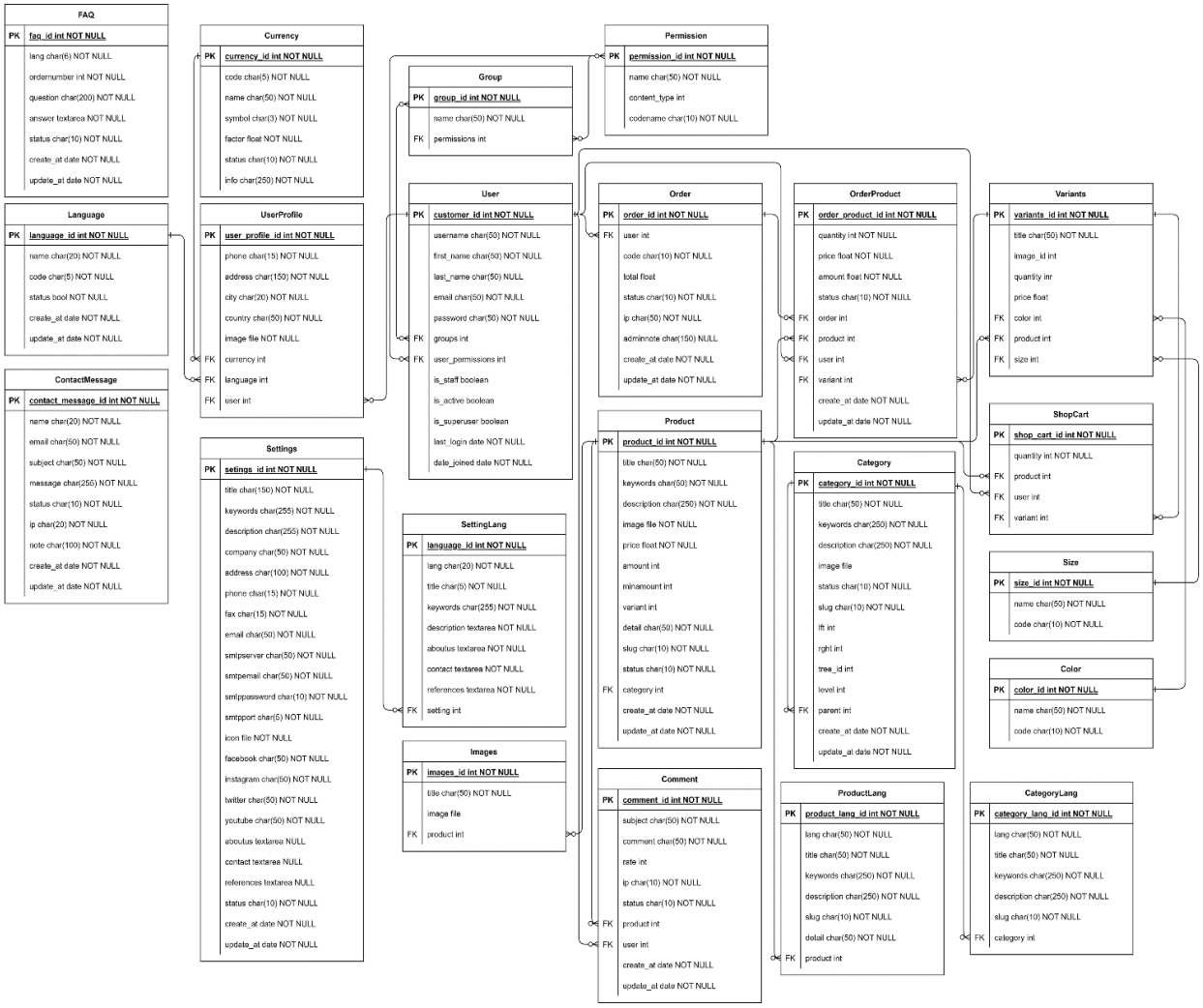

Рисунок 3.12 – База даних інтернет магазину

Основна таблиця – «Product», тому що вона буде використовуватись найчастіше, при оформлені замовлення. З нею користувач буде взаємодіяти найбільше.

Таблиця «FAQ» відповідає за часто запитувані питання, дана таблиця є важливою для користувачів у вирішенні проблем які в них можуть виникати.

Таблиці «Language», «ProductLang», «CategoryLang» та «SettingsLang» відповідають за зміну мови на сайті.

Таблиця «ContactMessage» відповідає за питання користувачів до адміністрації сайту, тут користувачі можуть запитувати конкретні питання які не знайдуть у таблиці «FAQ».

Таблиця «Currency» відповідає за вибір валюти на сайті, так же як і з мовою користувач може розраховуватись валютою якою забажає, це дозволить охопити більшу кількість користувачів.

Таблиці «User» та «UserProfile» відповідають за особисту інформацію користувачів. Також сюди можна віднести таблиці «Group» та «Permission» які відповідають за розділення прав доступу користувачів на сайті.

Таблиця «Settings» відповідає за збереження налаштувань сайту.

Таблиці «Order» та «OrderProduct» відповідає за замовлення які здійснє користувач. Таблиця «OrderProduct» об’єднує таблиці «Order» та «Product» зв’язком багато до багатьох.

Таблиця «Category» відповідає за збереження категорій товарів та потрібна для пошуку товарів по категоріям. Вона є багаторівнева, тобто категорія може мати багато підкатегорій.

Вежливими таблицями для товарів є таблиці «Size», «Color» та «Variants» вони відповідають за різні варіації комплектування товарів. Сюди можна віднести також таблицю, яка також повязана з товарами «Comment» яка відповідає за коментарі та оцінку користувачів на даний товар.

Також в товару може бути багато зображень, для детального огляду за це відповідає таблиця «Images».

3.3 Тестування та налагодження інформаційно-комп’ютерної системи.

Тестування Python Django зазвичай виконується за допомогою модуля unittest, який входить до складу стандартної бібліотеки Python. Django також надає власні засоби для тестування, такі як класи TestCase та Client для виконання тестових запитів до веб-додатка.

Основні кроки тестування Django виглядають наступним чином:

– створити класи, які успадковуються від django.test.TestCase і містять методи для перевірки очікуваної поведінки коду. У цих методах потрібно

викликати функції та методи Django, а також виконувати різні перевірки за допомогою методів assert;

– Django надає спеціальну базу даних для тестування, що дозволяє виконувати тести на ізольованій копії бази даних. Цю базу даних можна налаштувати у файлі конфігурації settings.py;

– запустити тести можна використовуючи команду «python manage.py test» у командному рядку. Django знайде всі тести і виконає їх;

– після завершення тестування Django повідомить про результати. Можна буде переглянути скільки тестів пройшли успішно, а також будь-які помилки чи відхилення.

Можна створити спеціальну директорію для тестування, але Django створює спеціальний файл test.py при створені нового додатку, в якому і будуть описані тести які має пройти система. При успішному проходженні тестувань Django виведе в термінал наступне повідомлення (рис. 3.13).

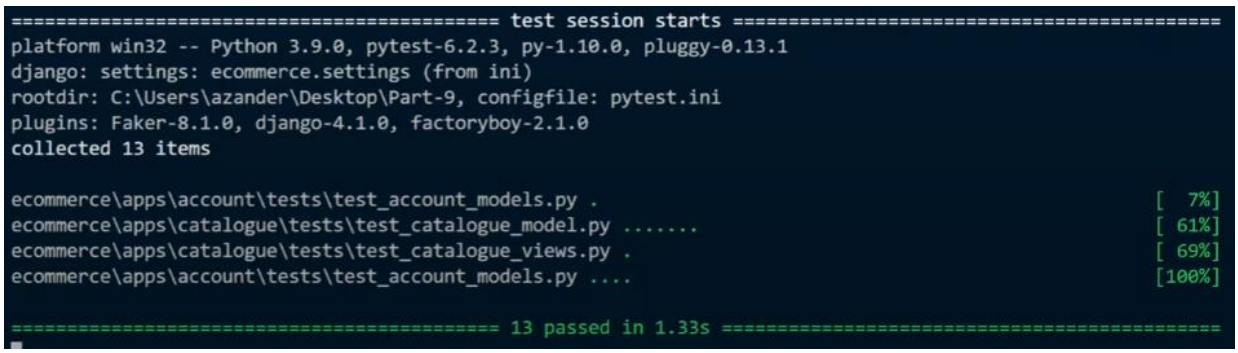

Рисунок 3.13 – Успішне проходження тестування Django

При невдалому проходженні тестування Django виведе в термінал наступне повідомлення з вказівкою де виникла помилка або не співпадіння очікуваних результатів (рис. 3.14).

```
        with pytest.raises(ValueError) as e:
            test = customer_factory.create(email="a.com")
>       assert str(e.value) == "You must provide a valid email addresss"
E       AssertionError: assert 'You must pro...email address' == 'You must pro...mail addresss'
E         - You must provide a valid email addresss
E         ?                                       -
E         + You must provide a valid email address

ecommerce\apps\account\tests\test_account_models.py:21: AssertionError
========================================= short test summary info =========================================
FAILED ecommerce/apps/account/tests/test_account_models.py::test_customer_email_incorrect - AssertionError:...
=================================== 1 failed, 10 passed in 1.38s ====================================
```

Рисунок 3.14 – Помилка або не співпадіння проходження тестування Django

Також можна проводити іншого роду тестування у самому браузері встановивши у Google Chrome режим як Low-end mobile, можна протестувати сайт у складних умовах – повільний інтернет та повільний смартфон з низьким розширенням (рис. 3.15).

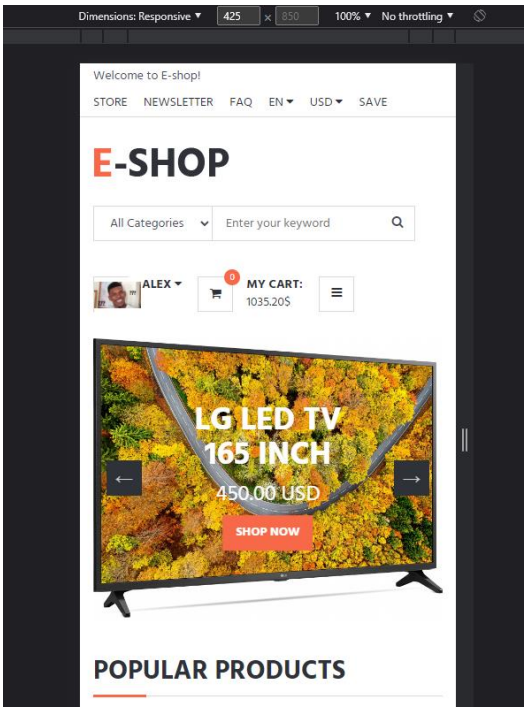

Рисунок 3.15 – Тестування на маленьких екранах

3.4 Розробка документації для програмного продукту

Документація надає опис програмного продукту – інтернет-магазину, розробленого на мові програмування Python 3.8 з використанням фреймворку

Django і бази даних PostgreSQL. Інтернет-магазин дозволяє користувачам переглядати товари, робити замовлення та здійснювати оплату. Документація надає вичерпну інформацію про функціональність, налаштування та розгортання програмного продукту.

Перед встановленням програмного продукту переконайтеся, що ваша система відповідає наступним вимогам:

– Python 3.8 або вище;
– Django 3.2 або вище;
– PostgreSQL 9.5 або вище;
– інтернет-підключення для зовнішнього доступу до платіжних шлюзів.

Для встановлення інтернет-магазину на своєму сервері виконайте наступні кроки:

a) склонуйте репозиторій з програмним кодом інтернет-магазину на свій локальний комп'ютер або сервер;

b) встановіть Python 3.8 або вище, якщо він ще не встановлений;

c) створіть віртуальне середовище і активуйте його;

d) встановіть залежності, перейшовши до кореневої папки проекту і виконавши наступну команду (лістинг 3.1):

Лістинг 3.1 – Команда встановлення залежностей

```
pip install -r requirements.txt
```

Кінець лістингу 3.1

e) створіть базу даних PostgreSQL і налаштуйте підключення до неї у файлі налаштувань Django (settings.py);

f) виконайте міграції для створення необхідних таблиць у базі даних (лістинг 3.2):

Лістинг 3.2 – Команда запуску міграцій

```
python manage.py migrate
```

Кінець лістингу 3.2

g) запустіть сервер Django (лістинг 3.3):

Лістинг 3.3 – Команда запуску сервера

```
python manage.py runserver
```

Кінець лістингу 3.3

h) відкрийте браузер і перейдіть за адресою http://localhost:8000 (або інша адреса, якщо ви налаштували інший порт або хост) для перегляду інтернет-магазину.

Інтернет-магазин може бути налаштований з використанням файлу налаштувань Django (settings.py). Основні параметри конфігурації включають:

DATABASES: Налаштування підключення до бази даних PostgreSQL.

STATIC_URL і STATIC_ROOT: Налаштування шляху до статичних файлів (CSS, JavaScript, зображення тощо).

MEDIA_URL і MEDIA_ROOT: Налаштування шляху до медіафайлів (завантажені користувачами зображення, відео тощо).

EMAIL_BACKEND і EMAIL_HOST: Налаштування електронної пошти для відправки повідомлень підтвердження замовлень і сповіщень.

Інтернет-магазин надає такі функції:

- зеєстрація та авторизація користувачів;
- перегляд списку доступних товарів;
- пошук товарів за категоріями, ключовими словами тощо;
- додавання товарів до кошика;
- оформлення замовлення з вказанням доставки та платіжних даних;
- відтвердження замовлення через електронну пошту;
- оплата замовлення через зовнішні платіжні шлюзи (наприклад, Stripe, PayPal і т.д.);
- відстеження стану замовлення користувачем;

– адміністративний доступ до панелі керування, де адміністратор може додавати, видаляти або редагувати товари, категорії, замовлення тощо.

Документація надає вичерпну інформацію про функціональність, встановлення та конфігурацію програмного продукту - інтернет-магазину. Цей документ допоможе розробникам та користувачам краще розуміти та використовувати програмний продукт у своїх проектах.

3.5 Розробка інсталяційного пакета

Щоб завантажити інтернет-магазин, написаний на Python Django з використанням бази даних PostgreSQL, на хостинг, потрібно виконати наступні кроки наступні кроки.

Переконатися, що хостинг підтримує Python Django та PostgreSQL. Для цього потрібно перевірити документацію обраного хостинг-провайдера або зв'яжзатися з їхньою підтримкою, щоб переконатися, що ці технології доступні на обраному плані.

Потрібно зберегти усі файли інтернет-магазину в одну директорію. Включаючи файли Django-проекту, включаючи manage.py, settings.py та інші необхідні файли.

Зберегти також файл зі залежностями проекту. Це може бути requirements.txt, який містить перелік пакетів Python, необхідних для роботи проекту. Якщо файл відсутній його можна створити за допомогою команди «`pip freeze > requirements.txt`».

Далі потрібно увійти в обліковий запис та перейти до панелі керування обраного хостинг-акаунту та зареєструвати ім’я домену, після чого створити базу даних PostgreSQL. Важливо записати дані для підключення до бази даних, такі як назва бази даних, ім'я користувача та пароль.

Також потрібно відредагувати файл налаштувань settings.py Django-проекту, щоб вказати налаштування підключення до бази даних PostgreSQL. Перейдемо до

розділу DATABASES у файлі settings.py та змінимо наступні параметри (лістинг 3.4).

Лістинг 3.4 – Налаштування підключення до бази даних

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'назва_бази_даних',
        'USER': 'ім\'я_користувача',
        'PASSWORD': 'пароль',
        'HOST': 'адреса_хоста',
        'PORT': 'порт',
    }
}
```

Кінець лістингу 3.4

Потрібно замінити поля 'назва_бази_даних', 'ім'я_користувача', 'пароль', 'адреса_хоста' та 'порт' відповідно до налаштувань підключення до бази даних PostgreSQL.

Також потрібно змінити у файлі settings.py налаштування дозволених хостів та вимкнути режим відладки або розробки. Параметр DEBUG потрібно перемкнути в значення False, а параметру ALLOWED_HOSTS дозволити всі хости (лістинг 3.5).

Лістинг 3.5 – Налаштування підключення до бази даних

```
DEBUG = False

ALLOWED_HOSTS = ['*']
```

Кінець лістингу 3.5

Далі використовуючи один з перерахованих варіантів такі як Git, FTP або SSH завантажимо файли проекту та файл requirements.txt на хостинг. У більшості випадків, можна завантажити файли в кореневу директорію хостинг-аккаунту або в папку, призначену для веб-додатків.

Для варіанту завантаження з Git потрібно мати створений аккаунт та відповідний репозиторій з файлами усього проекту (рис. 3.16).

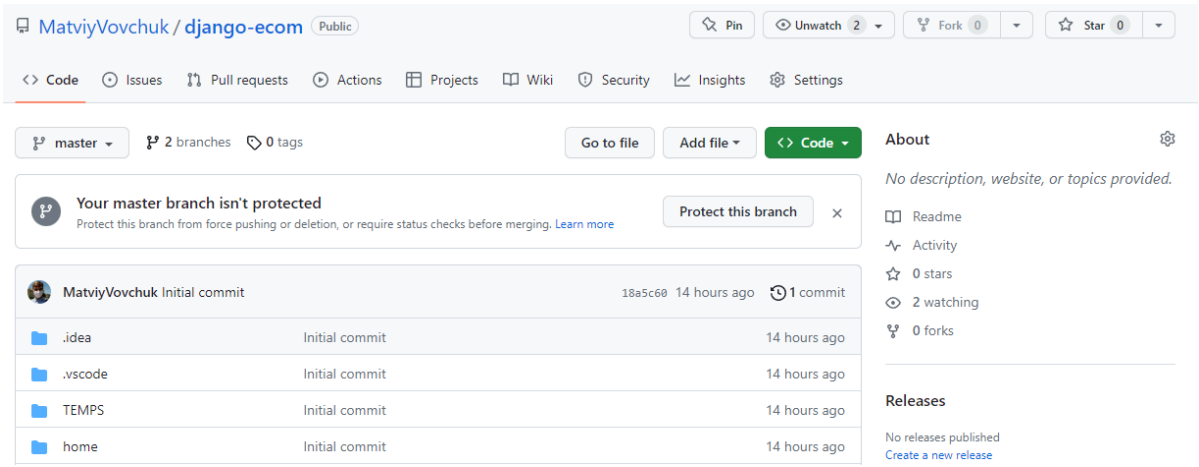

Рисунок 3.16 – Репозиторій з файлами проекту

Після завантаження файлів потрібно відкрити командний рядок або термінал на хостинг-акаунті та перейдемо до директорії, де розташовано Django-проект.

Якщо планується в подальшому завантажувати ще Python проекти рекомендується встановити віртуальне середовище для конкретного проекту, це можна зробити командою «`python -m venv *назва_віртуального_середовища*`» та активувати це середовище командою «`.\*назва_віртуального_середовища*\Scripts\activate`».

Після чого потрібно виконати команду «`pip install -r requirements.txt`» для встановлення всіх необхідних залежностей проекту.

Далі потрібно виконати команду «`python manage.py migrate`» для застосування міграцій до створеної бази даних PostgreSQL. Це створить таблиці, необхідні для Django-проекту.

Запустимо сервер Django, виконавши команду «`python manage.py runserver`». Ця команда запустить інтернет-магазин на веб-сервері.

Успішно виконавши ці кроки завантажимо розроблений інтернет-магазин на хостинг. Тепер можемо перевірити, чи працює інтернет-магазин, відкривши його URL-адресу в веб-браузері.

Висновки до розділу 3

Була описана та представлена практична реалізація об’єкту дослідження. Описана розроблена база даних, включаючи функціонування і призначення кожної з таблиць. Розроблена система була протестована написаними юніт тестами. Була розроблена документація та описаний процес розгортання системи на хостинг сервісі.

ВИСНОВКИ

У даній кваліфікаційній роботі було проведено розробку інтернет-магазину технічних товарів на мові Python з використанням таких технологій, як Django, PostgreSQL, JavaScript, HTML і CSS. Отримані результати та висновки підтверджують успішну реалізацію поставлених завдань та досягнення поставлених цілей проекту.

Одним із головних досягнень цієї роботи є створення повноцінного інтернет-магазину, здатного надати користувачам можливість переглядати, шукати та замовляти технічні товари. Розроблений застосунок забезпечує широкий функціонал, включаючи створення облікових записів користувачів, каталог товарів, корзину покупок та систему оплати. Крім того, інтерфейс магазину був розроблений з урахуванням сучасних тенденцій дизайну, що забезпечує зручну та привабливу взаємодію з користувачами.

Використання фреймворку Django дозволило значно спростити розробку магазину, забезпечивши гнучкість та швидкість в роботі з базою даних. Django надав можливість легко створювати моделі даних, керувати URL-шляхами, автоматично генерувати адміністративний інтерфейс та забезпечити безпеку додатку. Також використання PostgreSQL як системи керування базами даних забезпечило ефективну та надійну збереження інформації про товари, користувачів та замовлення.

JavaScript, HTML і CSS були використані для створення динамічного та привабливого інтерфейсу магазину. За допомогою JavaScript було реалізовано функціонал, який забезпечує взаємодію з користувачами на стороні клієнта, такий як додавання товарів до кошика, фільтрація товарів за різними критеріями та валідація введених даних. HTML та CSS дозволили створити структуру та стилізувати інтерфейс магазину, що забезпечує його зручне використання та привабливий зовнішній вигляд.

У процесі розробки інтернет-магазину було виявлено деякі труднощі, зокрема, складність налаштування деяких аспектів фреймворку Django та вирішення конфліктів між різними компонентами проекту. Однак завдяки глибокому вивченню документації та залученню додаткових ресурсів, ці проблеми було успішно вирішено.

У майбутньому розроблений інтернет-магазин може бути покращений та розширений шляхом додавання додаткового функціоналу, такого як система рекомендацій, інтеграція зі сторонніми сервісами оплати та доставки, покращення швидкодії та оптимізація інтерфейсу. Крім того, можна розглянути можливість підтримки багатомовності та мобільної версії магазину для забезпечення більш широкого кола користувачів.

У цілому, розробка інтернет-магазину технічних товарів на мові Python з використанням Django, PostgreSQL, JavaScript, HTML і CSS була успішною. Отримані результати свідчать про ефективне використання вищезгаданих технологій для розробки функціонального та зручного використання інтернет-магазину.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Python документація. URL: https://www.python.org/doc/ (дата звернення 06.03.2023).
2. Django документація. URL: https://docs.djangoproject.com/en/4.2/ (дата звернення 06.03.2023).
3. PostgreSQL документація. URL: https://www.postgresql.org/docs/ (дата звернення 06.03.2023).
4. Про MVC патерн. URL: https://en.wikipedia.org/wiki/Model–view–controller (дата звернення 10.04.2023).
5. Visual Studio Code in Action. URL: https://code.visualstudio.com/docs (дата звернення 12.02.2023).
6. Feldroy D., Feldroy A. Two Scoops of Django: Best Practices for Django 3.x, 2020. 485 с. (Two Scoops Press).
7. E-commerce Website using Django. URL: https://www.geeksforgeeks.org/e-commerce-website-using-django/ (дата звернення 21.02.2023).
8. Building Future Web Apps With JavaScript and Django. URL: https://medium.com/fbdevclagos/building-future-web-apps-with-javascript-and-django-c831883b22cf (дата звернення 23.03.2023).
9. Django, and the story of a CRSF cookie. URL: https://medium.com/@polyglot_factotum/django-and-the-story-of-a-crsf-cookie-b9384baf2db1 (дата звернення 03.04.2023).
10. Введення в SQL. URL: https://www.w3schools.com/sql/sql_intro.asp (дата звернення 10.03.2023).
11. Mangabo K. Full Stack Django and React: Get hands-on experience in full-stack web development with Python, React, and AWS, 2023. 432 с.
12. Lesley T. Web Developer: Learn how to create web applications in Python using frameworks like Django, Flask, and Pyramid, 2023. 70 с.
13. Django ORM і QuerySets. URL: https://tutorial.djangogirls.org/uk/django_orm/ (дата звернення 15.03.2023).

14. Flanagan D. JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language, 2020. 704 с.
15. Building eCommerce with Django Python Part 1. URL: https://medium.com/@bastianhilton/building-ecommerce-with-django-python-part-1-c4123746f7bf (дата звернення 16.04.2023).
16. Melé A. Django 3 By Example: Build powerful and reliable Python web applications from scratch, 3rd Edition, 2020. 568 с.
17. S. Vincent W. Django for Professionals: Production websites with Python & Django, 2020. 314 с.

ДОДАТКИ

Додаток А

Лістинг 1 – Юніт тесту для моделі Product

```
from django.test import TestCase
from product.models import Comment, Product, Category
from django.contrib.auth.models import User

class ProductTestCase(TestCase):
    @classmethod
    def setUpTestData(cls):
        # Створюємо категорію для тестування
        category = Category.objects.create(name='Test Category')

        # Створюємо продукт для тестування
        product = Product.objects.create(
            category=category,
            title='Test Product',
            keywords='Test',
            description='Test Description',
            price=10,
            amount=5,
            minamount=3,
            variant='None',
            slug='test-product',
            status='True'
        )

        # Створюємо користувача для перевірки методу avaregereview
        user = User.objects.create(username='testuser')

        # Створюємо коментарі для продукту
        Comment.objects.create(
            product=product,
            user=user,
            content='Test comment 1',
            rate=4,
            status='True'
        )
        Comment.objects.create(
            product=product,
            user=user,
            content='Test comment 2',
            rate=5,
            status='True'
        )

    def test_model_fields(self):
        # Отримуємо створений продукт для перевірки полів моделі
        product = Product.objects.get(title='Test Product')

        self.assertEqual(product.category.name, 'Test Category')
```

```
        self.assertEqual(product.keywords, 'Test')
        self.assertEqual(product.description, 'Test Description')
        self.assertEqual(product.price, 10)
        self.assertEqual(product.amount, 5)
        self.assertEqual(product.minamount, 3)
        self.assertEqual(product.variant, 'None')
        self.assertEqual(product.slug, 'test-product')
        self.assertEqual(product.status, 'True')

    def test_image_tag(self):
        # Отримуємо створений продукт для перевірки методу image_tag
        product = Product.objects.get(title='Test Product')

        # Перевіряємо, чи метод повертає очікуваний HTML-код
        expected_html = '<img src="{}"
height="50"/>'.format(product.image.url)
        self.assertEqual(product.image_tag(), expected_html)

    def test_get_absolute_url(self):
        # Отримуємо створений продукт для перевірки методу
get_absolute_url
        product = Product.objects.get(title='Test Product')

        # Перевіряємо, чи метод повертає очікувану URL-адресу
        expected_url = '/category/test-product/'
        self.assertEqual(product.get_absolute_url(), expected_url)

    def test_avaregereview(self):
        # Отримуємо створений продукт для перевірки методу
avaregereview
        product = Product.objects.get(title='Test Product')

        # Перевіряємо, чи метод повертає очікуваний середній рейтинг
        self.assertEqual(product.avaregereview(), 4.5)

    def test_countreview(self):
        # Отримуємо створений продукт для перевірки методу countreview
        product = Product.objects.get(title='Test Product')

        # Перевіряємо, чи метод повертає очікувану кількість
коментарів
        self.assertEqual(product.countreview(), 2)
```

Кінець лістингу 1

Лістинг 2 – Контролер додатку home

```
import json

from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.contrib.auth.forms import UserCreationForm
```

```
from django.contrib.auth.models import User
from django.db.models import Avg, Count, Q, F
from django.db.models.functions import Concat
from django.http import HttpResponse, HttpResponseRedirect,
JsonResponse, request
from django.shortcuts import render

# Create your views here.
from django.template.loader import render_to_string
from django.urls import reverse
from django.utils import translation

from home.forms import SearchForm
from home.models import Setting, ContactForm, ContactMessage, FAQ,
SettingLang, Language
from mysite import settings
from product.models import Category, Product, Images, Comment,
Variants, ProductLang, CategoryLang
from user.models import UserProfile


def index(request):
    if not request.session.has_key('currency'):
        request.session['currency'] = settings.DEFAULT_CURRENCY

    setting = Setting.objects.get(pk=1)
    products_latest = Product.objects.all().order_by('-id')[:4]  #
last 4 products
    # >>>>>>>>>>>>>>>> M U L T I   L A N G U G A E >>>>>> START
    defaultlang = settings.LANGUAGE_CODE[0:2]
    currentlang = request.LANGUAGE_CODE[0:2]

    if defaultlang != currentlang:
        setting = SettingLang.objects.get(lang=currentlang)
        products_latest = Product.objects.raw(
            'SELECT p.id,p.price, l.title, l.description,l.slug  '
            'FROM product_product as p '
            'LEFT JOIN product_productlang as l '
            'ON p.id = l.product_id '
            'WHERE  l.lang=%s ORDER BY p.id DESC LIMIT 4',
[currentlang])

    products_slider = Product.objects.all().order_by('id')[:4]
#first 4 products
    category = []

    products_picked = Product.objects.all().order_by('?')[:4]
#Random selected 4 products

    selectlanguage_url = reverse('selectlanguage')
    page="home"
    context={'setting':setting,
```

```
            'page':page,
            'products_slider': products_slider,
            'products_latest': products_latest,
            'products_picked': products_picked,
            'category': category,
            'selectlanguage_url': selectlanguage_url
            }
    return render(request,'index.html',context)



def selectlanguage(request):
    if request.method == 'POST':  # check post
        cur_language = translation.get_language()
        lasturl= request.META.get('HTTP_REFERER')
        lang = request.POST['language']
        translation.activate(lang)
        request.session[translation.LANGUAGE_SESSION_KEY]=lang
        # return HttpResponse(lang)

        redirect_url = "/" + request.POST['language']
        return JsonResponse({'redirect_url': redirect_url})
        # return HttpResponseRedirect("/" + lang)

def aboutus(request):
    #category = categoryTree(0,'',currentlang)
    defaultlang = settings.LANGUAGE_CODE[0:2]
    currentlang = request.LANGUAGE_CODE[0:2]
    setting = Setting.objects.get(pk=1)
    if defaultlang != currentlang:
        setting = SettingLang.objects.get(lang=currentlang)

    context={'setting':setting}
    return render(request, 'about.html', context)

def contactus(request):
    currentlang = request.LANGUAGE_CODE[0:2]
    #category = categoryTree(0,'',currentlang)
    if request.method == 'POST': # check post
        form = ContactForm(request.POST)
        if form.is_valid():
            data = ContactMessage() #create relation with model
            data.name = form.cleaned_data['name'] # get form input
data
            data.email = form.cleaned_data['email']
            data.subject = form.cleaned_data['subject']
            data.message = form.cleaned_data['message']
            data.ip = request.META.get('REMOTE_ADDR')
            data.save()  #save data to table
            messages.success(request,"Your message has ben sent.
Thank you for your message.")
```

```
            return HttpResponseRedirect('/contact')

    defaultlang = settings.LANGUAGE_CODE[0:2]
    currentlang = request.LANGUAGE_CODE[0:2]
    setting = Setting.objects.get(pk=1)
    if defaultlang != currentlang:
        setting = SettingLang.objects.get(lang=currentlang)

    form = ContactForm
    context={'setting':setting,'form':form  }
    return render(request, 'contactus.html', context)

def category_products(request,id,slug):
    defaultlang = settings.LANGUAGE_CODE[0:2]
    currentlang = request.LANGUAGE_CODE[0:2]
    catdata = Category.objects.get(pk=id)
    products = Product.objects.filter(category_id=id) #default
language
    if defaultlang != currentlang:
        try:
            products = Product.objects.raw(
                'SELECT
p.id,p.price,p.amount,p.image,p.variant,l.title, l.keywords,
l.description,l.slug,l.detail '
                'FROM product_product as p '
                'LEFT JOIN product_productlang as l '
                'ON p.id = l.product_id '
                'WHERE p.category_id=%s and l.lang=%s', [id,
currentlang])
        except:
            pass
        catdata = CategoryLang.objects.get(category_id=id,
lang=currentlang)

    context={'products': products,
             #'category':category,
             'catdata':catdata }
    return render(request,'category_products.html',context)

def search(request):
    if request.method == 'POST': # check post
        form = SearchForm(request.POST)
        if form.is_valid():
            query = form.cleaned_data['query'] # get form input data
            catid = form.cleaned_data['catid']
            if catid==0:

products=Product.objects.filter(title__icontains=query)  #SELECT *
FROM product WHERE title LIKE '%query%'
            else:
                products =
Product.objects.filter(title__icontains=query,category_id=catid)
```

```
            category = Category.objects.all()
            context = {'products': products, 'query':query,
                       'category': category }
            return render(request, 'search_products.html', context)

    return HttpResponseRedirect('/')

def search_auto(request):
    if request.is_ajax():
        q = request.GET.get('term', '')
        products = Product.objects.filter(title__icontains=q)

        results = []
        for rs in products:
            product_json = {}
            product_json = rs.title +" > " + rs.category.title
            results.append(product_json)
        data = json.dumps(results)
    else:
        data = 'fail'
    mimetype = 'application/json'
    return HttpResponse(data, mimetype)

def product_detail(request,id,slug):
    query = request.GET.get('q')
    # >>>>>>>>>>>>>>>> M U L T I   L A N G U G A E >>>>>> START
    defaultlang = settings.LANGUAGE_CODE[0:2] #en-EN
    currentlang = request.LANGUAGE_CODE[0:2]
    #category = categoryTree(0, '', currentlang)
    category = Category.objects.all()

    product = Product.objects.get(pk=id)

    if defaultlang != currentlang:
        try:
            prolang =  Product.objects.raw('SELECT
p.id,p.price,p.amount,p.image,p.variant,l.title, l.keywords,
l.description,l.slug,l.detail '
                                           'FROM product_product as p
'
                                           'INNER JOIN
product_productlang as l '
                                           'ON p.id = l.product_id '
                                           'WHERE p.id=%s and
l.lang=%s',[id,currentlang])
            product=prolang[0]
        except:
            pass
    # <<<<<<<<<< M U L T I   L A N G U G A E <<<<<<<<<<<<<< end

    images = Images.objects.filter(product_id=id)
```

```
    comments = Comment.objects.filter(product_id=id,status='True')
    context = {'product': product,'category': category,
               'images': images, 'comments': comments,
               }
    if product.variant !="None": # Product have variants
        if request.method == 'POST': #if we select color
            variant_id = request.POST.get('variantid')
            variant = Variants.objects.get(id=variant_id) #selected
product by click color radio
            colors =
Variants.objects.filter(product_id=id,size_id=variant.size_id )
            sizes = Variants.objects.raw('SELECT * FROM
product_variants  WHERE product_id=%s GROUP BY size_id',[id])
            query += variant.title+' Size:' +str(variant.size) +'
Color:' +str(variant.color)
        else:
            variants = Variants.objects.filter(product_id=id)
            colors =
Variants.objects.filter(product_id=id,size_id=variants[0].size_id )
            sizes = Variants.objects.raw('SELECT * FROM
product_variants  WHERE product_id=%s GROUP BY size_id',[id])
            variant =Variants.objects.get(id=variants[0].id)
        context.update({'sizes': sizes, 'colors': colors,
                        'variant': variant,'query': query
                        })
    return render(request,'product_detail.html',context)

def ajaxcolor(request):
    data = {}
    if request.POST.get('action') == 'post':
        size_id = request.POST.get('size')
        productid = request.POST.get('productid')
        colors = Variants.objects.filter(product_id=productid,
size_id=size_id)
        context = {
            'size_id': size_id,
            'productid': productid,
            'colors': colors,
        }
        data = {'rendered_table':
render_to_string('color_list.html', context=context)}
        return JsonResponse(data)
    return JsonResponse(data)


def faq(request):
    defaultlang = settings.LANGUAGE_CODE[0:2]
    currentlang = request.LANGUAGE_CODE[0:2]

    if defaultlang==currentlang:
```

```
        faq =
FAQ.objects.filter(status="True",lang=defaultlang).order_by("ordernu
mber")
    else:
        faq =
FAQ.objects.filter(status="True",lang=currentlang).order_by("ordernu
mber")

    context = {
        'faq': faq,
    }
    return render(request, 'faq.html', context)


def selectcurrency(request):
    lasturl = request.META.get('HTTP_REFERER')
    if request.method == 'POST':  # check post
        request.session['currency'] = request.POST['currency']
    return HttpResponseRedirect(lasturl)

@login_required(login_url='/login') # Check login
def savelangcur(request):
    lasturl = request.META.get('HTTP_REFERER')
    curren_user = request.user
    language=Language.objects.get(code=request.LANGUAGE_CODE[0:2])
    #Save to User profile database
    data = UserProfile.objects.get(user_id=curren_user.id )
    data.language_id = language.id
    data.currency_id = request.session['currency']
    data.save()  # save data
    return HttpResponseRedirect(lasturl)
```

Кінець лістингу 2