Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Автоматизована система з підбору комп'ютерних комплектуючих»

Виконав: студент 2 курсу, групи 1АКІТР-24м
спеціальності 174 – Автоматизація,
комп'ютерно-інтегровані технології та
робототехніка
(шифр і назва спеціальності)
Олександр ГОРДІЄНКО
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. АІІТ
Марія БАРАБАН
(науковий ступінь, вчене звання / посада, ПІБ керівника)
«12» 12 2025 р.

Опонент: д.т.н., проф. каф. АІІТ
Володимир ОЗЕРАНСЬКИЙ
(науковий ступінь, вчене звання / посада, ПІБ опонента)
«12» 12 2025 р.

Допущено до захисту

Завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
(прізвище та ініціали)
«15» 12 2025 р.

Вінниця ВНТУ - 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 17 «Електроніка, автоматизація та електронні комунікації»
Спеціальність – 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»
Освітньо-професійна програма – «Інтелектуальні комп'ютерні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
проф., д.т.н. Олег БІСІКАЛО
"26" 09 2025 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Гордієнку Олександру Олександровичу

1.Тема роботи: «Автоматизована система з підбору комп'ютерних комплектуючих».
Керівник роботи: Барабан Марія Володимирівна, к.т.н., доцент.
Затверджені наказом ВНТУ від «24» вересня 2025 року № 313.
2. Строк подання роботи студентом: до «12» грудня 2025 року.
3.Вихідні дані до роботи:кількість користувачів – до 100чол;кількість комплектуючих для аналізу – не менше 10шт;імова програмування – об'єктно-орієнтована.
4.Зміст текстової частини:вступ; аналіз сучасних технологій для підбору комп'ютерних комплектуючих; моделювання автоматизованої системи підбору комп'ютерних комплектуючих; програмна реалізація автоматизованої системи підбору комп'ютерних комплектуючих; тестування та аналіз результатів роботи розробленої системи; висновки, перелік використаних джерел; додатки.
5. Перелік ілюстративного (або графічного) матеріалу: схема алгоритму функціонування автоматизованої системи підбору комп'ютерних комплектуючих; вигляд інтерфейсу користувача автоматизованої системи підбору комп'ютерних комплектуючих; приклади роботи програм

6.Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Марія БАРАБАН, к.т.н., доц. каф. АІІТ	25.09.2025	03.12.2025
4	Володимир КОЗЛОВСЬКИЙ, к.е.н., проф. каф. ЕПтаВМ	25.09.2025	01.12.2025

7.Дата видачі завдання: «25» вересня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Прим
1	Аналіз сучасних технологій підбору комп'ютерних комплектуючих	25.09 – 05.10.2025	Виконано
2	Моделювання автоматизованої системи підбору комп'ютерних комплектуючих	05.10 – 15.10.2025	Виконано
3	Програмна реалізація автоматизованої системи підбору комп'ютерних комплектуючих	15.10 – 31.10.2025	Виконано
4	Тестування та аналіз результатів роботи розробленої системи	01.11 – 10.11.2025	Виконано
5	Розробка інструкції користувача	10.11 – 20.11.2025	Виконано
6	Підготовка економічної частини	до 01.12.2025	Виконано
7	Оформлення пояснювальної записки, графічного матеріалу і презентації	20.10 – 28.11.2025	Виконано
8	Попередній захист магістерської роботи	до 03.12.2025	Виконано
9	Захист магістерської роботи	до 19.12.2025	Виконано

Студент _____ Олександр ГОРДІЄНКО
(підпис) (прізвище та ініціали)

Керівник роботи _____ Марія БАРАБАН
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.738.5:004.774.6

Гордієнко О.О. Автоматизована система з підбору комп'ютерних комплектуючих. Магістерська кваліфікаційна робота зі спеціальності 174 – Автоматизація, комп’ютерно-інтегровані технології та робототехніка, освітня програма – Інтелектуальні комп’ютерні системи. Вінниця: ВНТУ, 2025. 119 с.

Укр. мовою. Бібліогр.: 25 назв; рис.: 20; табл 8.

У роботі розглянуто питання проєктування та розробки автоматизованої системи підбору комп'ютерних комплектуючих з використанням сучасних веб-технологій та принципів електронної комерції. Проведено аналіз існуючих конфігураторів ПК та визначено основні критерії якісного підбору компонентів з урахуванням сумісності, енергетичного балансу та відсутності вузьких місць у системі. Запропоновано багаторівневу архітектуру веб-платформи, що інтегрує рівні презентації, бізнес-логіки, даних та зовнішніх сервісів, забезпечує мультирольовий доступ, автоматизовану перевірку сумісності компонентів та безпечну обробку платіжних транзакцій через Stripe. Виконано моделювання системи засобами UML з розробкою діаграм класів, послідовності та структурної схеми. Реалізовано функціональний прототип на базі Django framework з інтеграцією PostgreSQL, AWS S3, Bootstrap та інших сучасних технологій. Проведено комплексне тестування розробленої системи, що підтвердило її повну працездатність, надійність та готовність до комерційного використання.

Ключові слова: автоматизована система підбору комплектуючих, електронна комерція, веб-платформа, сумісність компонентів, інтеграція платіжних систем, багаторівнева архітектура, моделювання UML, Django framework, онлайн-конфігуратор ПК.

ABSTRACT

Hordiienko O.O. Automated system for selecting computer components. Master's thesis in specialty 174 – Automation, computer-integrated technologies and robotics, educational program – Intelligent Computer Systems. Vinnytsia: VNTU, 2025. 119 p.

In Ukrainian language. Bibliography: 25 titles; Fig.: 20; table 8.

The thesis addresses the design and development of an automated system for selecting computer components using modern web technologies and e-commerce principles. An analysis of existing PC configurators was conducted, and key criteria for quality component selection were identified, considering compatibility, energy balance, and elimination of system bottlenecks. A multi-tier web platform architecture is proposed, integrating presentation, business logic, data, and external services layers, providing multi-role access, automated component compatibility checking, and secure payment transaction processing through Stripe. System modeling was performed using UML with development of class diagrams, sequence diagrams, and structural schemas. A functional prototype was implemented based on Django framework with integration of PostgreSQL, AWS S3, Bootstrap, and other modern technologies. Comprehensive testing of the developed system was conducted, confirming its full functionality, reliability, and readiness for commercial use.

Keywords: automated component selection system, e-commerce, web platform, component compatibility, payment system integration, multi-tier architecture, UML modeling, Django framework, online PC configurator.

ЗМІСТ

ВСТУП

Актуальність теми. Індивідуальне збирання комп’ютера стало однією з найпоширеніших практик серед сучасних користувачів, які прагнуть максимально ефективно витратити кошти при придбанні комп’ютерної техніки. Такий підхід дозволяє самостійно обирати компоненти, які найкраще відповідають потребам конкретного користувача — чи то для офісної роботи, ігор, професійної діяльності (наприклад, графічного дизайну чи програмування), чи для домашнього використання. Саме можливість досягти найкращого балансу між ціною, продуктивністю та функціональністю робить індивідуальне збирання комп’ютера надзвичайно привабливим.

Однак такий підхід вимагає від користувача ґрунтовних знань у ряді спеціалізованих предметних областей. Зокрема, необхідно розуміти основи електроніки, принципи схемотехніки, особливості архітектури комп’ютерів, характеристик і стандартів комп’ютерних інтерфейсів (наприклад, сокети процесорів, типи пам’яті, інтерфейси накопичувачів тощо). Без належної обізнаності у цих питаннях існує високий ризик придбання несумісних комплектуючих, що не лише ускладнює процес збирання, але й може призвести до додаткових витрат часу і коштів.

Більше того, навіть досвідчені користувачі, які мають значний досвід конструювання комп’ютерної техніки, часто стикаються з проблемами несумісності апаратних компонентів. Наприклад, можуть виникнути ситуації, коли обраний процесор не підтримується материнською платою через відмінності в чипсетах, або коли блок живлення не забезпечує достатньої потужності для обраної конфігурації. Іноді навіть неочевидні чинники, такі як фізичні розміри корпусу чи система охолодження, можуть стати критичними при збиранні системи.

У зв’язку з цим надзвичайно актуальною є задача створення автоматизованої системи, яка б дозволяла автоматизувати процес підбору комп’ютерних комплектуючих. Така система могла б брати до уваги параметри сумісності між різними елементами, поточні ринкові ціни, а також специфічні вимоги користувача. Вона допоможе значно зменшити вплив людського фактора, уникнути типових помилок при підборі компонентів, а також суттєво скоротити час, необхідний на формування оптимальної конфігурації комп’ютера. Крім того, використання такої інформаційної системи стане корисним інструментом як для новачків, так і для професіоналів, яким важливо швидко отримати перевірене й ефективне рішення.

Метою роботи є удосконалення системи підбору комп’ютерних комплектуючих для підвищення ефективності та якості складання комп’ютерної техніки за рахунок розробки та впровадження автоматизованої системи, що забезпечить ефективну взаємодію з користувачами та оптимізує параметри обраної комп’ютерної техніки.

Для досягнення цієї мети потрібно вирішити наступні завдання:

1. Здійснити аналіз предметної області в галузі підбору комп’ютерних комплектуючих.

2. Провести огляд існуючих рішень та систем для автоматизованого підбору комп’ютерних комплектуючих.

3. Здійснити програмну реалізацію автоматизованої системи підбору комп’ютерних комплектуючих.

4. Провести тестування та порівняння з аналогами.

Об'єктом дослідження є процеси автоматизованого підбору комп’ютерних комплектуючих.

Предметом дослідження є програмні засоби автоматизованого підбору комп’ютерних комплектуючих.

Методи дослідження – збір даних про процеси підбору комп’ютерних комплектуючих, вимоги користувачів та показники продуктивності системи, вивчення існуючих досліджень та публікацій, пов'язаних з автоматизованими системами підбору комп’ютерних комплектуючих.

Науково-технічний результат полягає в удосконаленні системи підбору комп’ютерних комплектуючих для підвищення ефективності та якості складання комп’ютерної техніки за рахунок розробки та впровадження автоматизованої системи, що забезпечить ефективну взаємодію з користувачами та оптимізує параметри обраної комп’ютерної техніки.

Практична цінність полягає в створенні продуктивної автоматизованої системи підбору комп’ютерних комплектуючих. Ця система дозволить оптимізувати робочі процеси, забезпечуючи швидкий та якісний підбір необхідних комплектуючих комп’ютера для забезпечення оптимальності його роботи.

Апробація результатів дослідження. Результати роботи були апробовані на міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» (м. Вінниця, Україна, 2025 р.) [1].

Публікації. За результатами досліджень опубліковано тези доповіді на науково-практичній конференції [1].

1 АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ ПІДБОРУ КОМП'ЮТЕРНИХ КОМПЛЕКТУЮЧИХ

1.1 Дослідження аспектів складання списків комплектуючих для комп'ютера

Питання індивідуального складання персонального комп'ютера набуває особливої актуальності в умовах високих темпів розвитку інформаційних технологій та збільшення обсягу задач, що потребують оптимізованих апаратних рішень. Незважаючи на значну кількість готових конфігурацій, представлених на ринку, простежується тенденція до зростання інтересу користувачів щодо самостійного формування апаратної структури ПК. Такий підхід надає можливість здійснювати повний контроль над якістю компонентів, забезпечує гнучкість під час вибору технічних характеристик та дозволяє мінімізувати витрати шляхом виключення функціоналу, що не впливає на цільову продуктивність системи. Індивідуальна конфігурація не лише сприяє адаптації пристрою до специфіки поставлених завдань, але й поглиблює розуміння його внутрішньої архітектури, що має важливе значення для користувачів-початківців та ентузіастів.

Самозбірка персонального комп'ютера розглядається не лише як спосіб економії фінансових ресурсів, але й як інструмент забезпечення відповідності кінцевого пристрою функціональним потребам користувача. У разі придбання готової системної одиниці існує висока ймовірність переплати за бренд або за наявність опцій, що не застосовуються у реальному використанні. Натомість індивідуальний підбір компонентів дозволяє орієнтуватися на конкретні вимоги: ігровий сегмент потребує високопродуктивної графічної підсистеми, інженерні та дизайнерські завдання — потужного центрального процесора, а офісні конфігурації — переважно низького рівня шуму й стабільності роботи.

Важливо й те, що користувач повністю обізнаний щодо апаратної комплектації, що унеможливлює використання маркетингових маніпуляцій та сприяє підвищенню операційної надійності.

Перед початком процесу формування апаратної конфігурації доцільно визначити структурні елементи, які забезпечують функціонування ПК. Серед ключових компонентів виокремлюють: материнську плату, центральний процесор, оперативну пам’ять, пристрої зберігання даних, джерело живлення, корпус та дискретну відеокарту (у випадку відсутності інтегрованої). Необхідною умовою є структурно-функціональна сумісність елементів. Так, тип процесорного сокету має відповідати форм-фактору та чипсету материнської плати, а потужність блока живлення — перевищувати сумарну енергоспоживальність компонентів із технологічним резервом. Відповідно до рекомендацій досвідчених користувачів, початковий етап оптимальної конфігурації доцільно здійснювати з вибору процесора, оскільки саме він визначає технологічний рівень системи. Подальший відбір компонентів виконується з урахуванням сумісності, фінансових обмежень, планованих обчислювальних навантажень, якості матеріалів, теплового пакету та акустичного комфорту.

Однією з найбільш поширених проблем серед користувачів-початківців є несумісність компонентів. Її прояв може включати невідповідність частот оперативної пам’яті, недостатність потужності джерела живлення для відеокарти або невідповідність процесора конкретному сокету. Такі помилки спричиняють додаткові витрати та зниження ефективності процесу складання. Тому доцільним є ретельний попередній аналіз технічних характеристик кожної складової. Не рекомендується орієнтуватися лише на зовнішню подібність компонентів, оскільки навіть модулі пам’яті одного стандарту можуть бути взаємно несумісні. Нерідко подібні упущення призводять до того, що система не запускається після складання. Використання спеціалізованих інструментів

для підбору апаратної конфігурації в онлайн-середовищі дозволяє значною мірою уникнути зазначених ризиків.

Сучасні конфігуратори комп’ютерних систем функціонують як інтелектуальні сервіси, що здійснюють автоматизований добір компонентів із урахуванням їхньої сумісності. Алгоритми таких платформ забезпечують рекомендації щодо супутніх компонентів відразу після вибору базового елемента (наприклад, процесора), що мінімізує ймовірність критичних відхилень. Крім того, сервіси містять відгуки користувачів, рейтингові оцінки та експертні поради, що є релевантним для користувачів з обмеженим бюджетом. Завдяки таким конфігураторам процес формування ПК доступний навіть за відсутності глибоких технічних знань, що робить їх ефективним інструментом як для новачків, так і для досвідчених фахівців.

Для значної кількості користувачів процес складання ПК трансформується в технічне хобі. Кожна стадія — від аналізу ринку компонентів до інсталяції програмного забезпечення — сприяє розвитку прикладних технічних навичок, логіко-аналітичного мислення та розуміння принципів функціонування сучасних комп’ютерних систем. Окрім потенційної економії, самостійне складання дозволяє сформувати високопродуктивну систему з оптимальним співвідношенням вартості та потужності. Водночас результатом є не лише технічна ефективність, але й суб’єктивне задоволення від реалізації власного проєкту, особливо за умови першого успішного запуску системи.

Ключовим чинником надійності ПК є збалансованість конфігурації. Орієнтація виключно на максимальні показники продуктивності може призвести до створення "вузьких місць", які обмежуватимуть потенціал компонентів. Наприклад, високопродуктивний центральний процесор не здатний забезпечити ефективну роботу в умовах недостатнього обсягу оперативної пам’яті або використання малопотужної відеокарти. Отже,

доцільно здійснювати структурну оптимізацію конфігурації відповідно до функціональних потреб: для ігрових систем — пріоритет на графічну підсистему, для робочих станцій — на стабільність, багатозадачність і теплову ефективність, для офісних — енергоощадність та компактність.

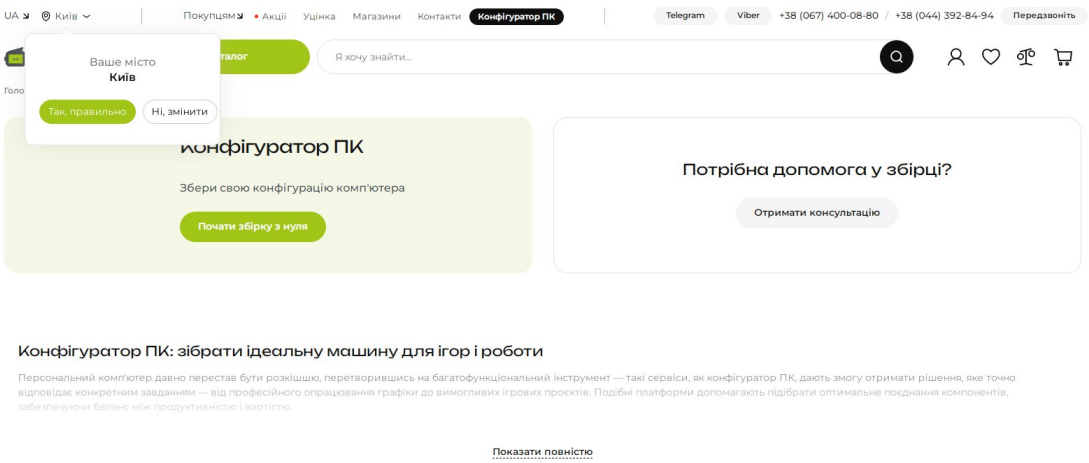
Цифрові технології значно спростили процес формування персонального комп’ютера. На сьогодні для складання ПК в онлайн-середовищі достатньо перейти на веб-платформу конфігуратора та послідовно обрати компоненти. Багато сервісів пропонують готові шаблони конфігурацій під різні сценарії використання (ігри, офіс, відеомонтаж), що скорочує час вибору та знижує ймовірність технічних похибок. Крім того, системи забезпечують автоматичний розрахунок вартості та порівняння з альтернативними компонентами, що дозволяє оптимізувати бюджет.

Особливо значущим аспектом є узгодження характеристик процесора та відеокарти. У випадку недостатньої потужності ЦП продуктивність GPU не реалізується у повному обсязі, що створює ефект "вузького місця". З метою уникнення таких ситуацій доцільно використовувати результати синтетичного тестування та аналіз реальних сценаріїв навантаження. Під час вибору також необхідно враховувати енергоспоживання та тепловиділення, оскільки високопродуктивні графічні адаптери потребують джерел живлення зі значною запасною потужністю та ефективної системи охолодження.

1.2 Аналіз програм аналогів для підбору комп’ютерних комплектуючих

Конфігуратор ПК від TELEMART – це онлайн-інструмент для самостійного підбору та перевірки сумісності комп'ютерних комплектуючих. Він дозволяє вибрати компоненти, такі як процесор, материнська плата, відеокарта, оперативна пам'ять, накопичувач, блок

живлення та корпус, а також автоматично перевіряє їх сумісність, запобігаючи помилкам у збірці (рис. 1.1).



Р

исунок 1.1 – Конфігуратор telemart

Ключові функції та переваги конфігуратора TELEMART:

˜ Вибір комплектуючих. Користувач може самостійно зібрати комп'ютер, обираючи кожен компонент із запропонованого списку.

˜ Автоматична перевірка сумісності. Конфігуратор перевіряє сумісність компонентів за різними параметрами, наприклад:

˜ Процесор та материнська плата: Перевірка сокету та версії BIOS.

˜ Процесор та кулер: Перевірка сумісності за тепловиділенням (TDP).

˜ Материнська плата та корпус: Перевірка форм-факторів.

˜ Кулер та корпус: Перевірка габаритів, зокрема висоти кулера.

˜ Оперативна пам'ять та материнська плата. Перевірка сумісності модулів пам'яті та слотів.

˜ Виявлення помилок. Інструмент інформує про потенційні проблеми та конфлікти між компонентами, дозволяючи їх усунути перед покупкою.

˜ Рекомендаційний характер. Конфігуратор має рекомендаційний характер і не обмежує користувача у процесі вибору, надаючи свободу дій.

˜ Підтримка апгрейду. Інструмент може бути корисним не тільки для повної збірки, але й для підбору комплектуючих для оновлення вже наявної системи.

Конфігуратор ПК COMPX — це інструмент, який дозволяє користувачам створювати комп'ютер, обираючи необхідні комплектуючі, щоб зібрати систему відповідно до своїх потреб. Він включає основні компоненти, такі як процесор, материнська плата, оперативна пам'ять (RAM), накопичувачі (SSD/HDD), відеокарта та блок живлення, що дає змогу гнучко підбирати параметри системи (рис. 1.2).

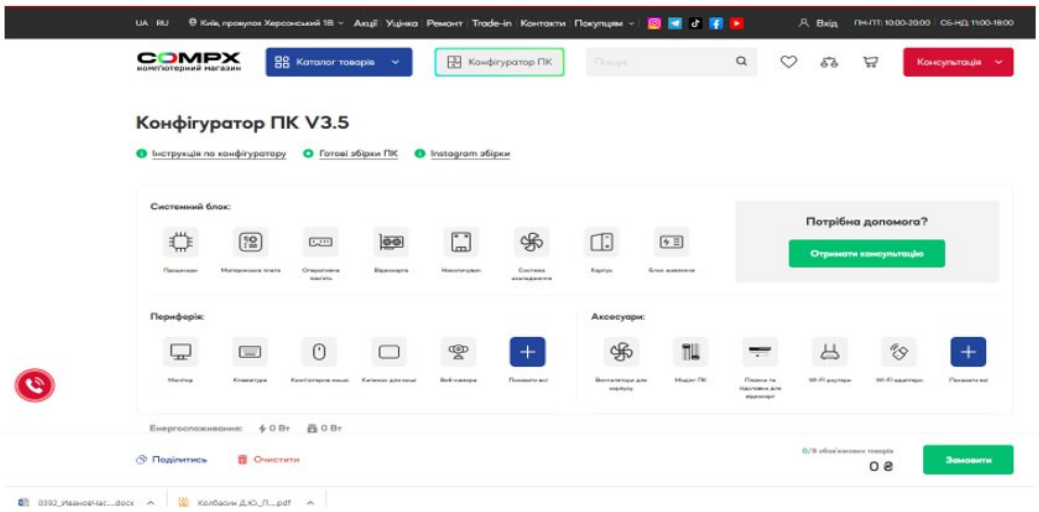


Рисунок 1.2 – Конфігуратор COMPX

Основні компоненти, які можна вибрати:

˜ Процесор (CPU): центральний обчислювальний блок ПК, який впливає на загальну продуктивність.

˜ Материнська плата: основа системи, яка з'єднує всі компоненти та забезпечує їхню взаємодію.

˜ Оперативна пам'ять (RAM): тимчасове сховище для даних, необхідних для роботи програм.

˜ Накопичувачі: SSD (твердотільний накопичувач): забезпечує швидке завантаження операційної системи та програм. HDD (жорсткий диск): використовується для тривалого зберігання великих обсягів даних.

˜ Відеокарта (GPU): обробляє графіку та відео, необхідна для ігор та професійних завдань.

˜ Блок живлення (PSU): забезпечує енергією всі компоненти системи.

˜ Система охолодження: відводить тепло від процесора та інших компонентів, забезпечуючи їх стабільну роботу.

1.3 Постановка задачі

Здійснимо постановку задачі на розробку автоматизованої системи з підбору комп’ютерних комплектуючих.

1. Функціональні вимоги

- Система аутентифікації та управління обліковими записами

Забезпечити реєстрацію користувачів з підтвердженням електронної пошти, авторизацію, вихід із системи, можливість відновлення пароля. Для авторизованих користувачів – доступ до профілю з редагуванням даних та перегляд історії замовлень.

- Навігація сайтом

Реалізувати адаптивну навігаційну панель з логотипом, яка змінює прозорість при прокручуванні сторінки. Меню динамічно змінюється залежно від статусу користувача (гість, авторизований, адміністратор).

- Управління товарами (каталог)

Забезпечити перегляд списку товарів, відображення карток з фото, назвою, ціною, рейтингом, категорією. Реалізувати пошук, фільтрацію за категоріями та параметрами.

- Сторінка товару

Відображення детальної інформації, наявності на складі, можливість додавання до кошика та список бажаного (wishlist), керування кількістю.

- Кошик та оформлення замовлення

Додати можливість перегляду доданих товарів, редагування кількості, видалення позицій. Під час оформлення – форма для введення контактних та платіжних даних із можливістю автозаповнення з профілю. Реалізувати систему онлайн-оплати через Stripe та підтвердження замовлення з надсиланням електронного листа.

- Система списку бажаного (Wishlist)

Доступна лише авторизованим користувачам, дозволяє додавання та видалення товарів.

- Адміністративний модуль

Адміністратор може додавати, редагувати та видаляти товари. Доступ – лише для авторизованих користувачів з роллю адміністратора. Перед видаленням – підтвердження через модальне вікно.

- Контактна форма

Передбачити можливість надсилання звернень користувачами. Після заповнення – повернення на головну сторінку з відповідним повідомленням.

- Підписка на розсилку

У футері забезпечити форму підписки на інформаційний бюлетень.

- Динамічне оновлення кошика

При додаванні товару – автоматичне оновлення іконки кошика з показом суми та зміною кольору.

-Мультимедійний дизайн головної сторінки

Додати фонове відео про геймінг та комп’ютерні компоненти, з адаптивною навігацією поверх.

-Можливість реалізації майбутніх функцій (як roadmap):

– форум з відповідями, обговореннями та лайками;

– повноцінна система контролю запасів зі сповіщенням при появі в наявності;

– кнопка Buy Now для швидкої покупки одного товару.

2. Нефункціональні вимоги

-Продуктивність

Система повинна забезпечувати швидке завантаження сторінок та коректну роботу при одночасному використанні значної кількості користувачів.

-Надійність та безпека

Використання захищених протоколів передачі даних (HTTPS), шифрування паролів, захист від SQL-ін’єкцій та атак типу XSS/CSRF. Платіжна система повинна відповідати стандартам PCI DSS.

-Масштабованість

Система має допускати можливість розширення функціоналу (форум, інвентаризація, push-сповіщення).

-Юзабіліті

Сайт повинен мати сучасний інтерфейс, інтуїтивну навігацію, адаптивний дизайн для мобільних і десктопних пристроїв. Всі повідомлення повинні бути інформативними та зрозумілими.

-Сумісність

Коректна робота веб-додатку у найпоширеніших браузерах (Chrome, Firefox, Edge, Safari) та на основних розширеннях екрана.

-Підтримуваність

Структура коду повинна бути модульною та відповідати принципам розширюваності. Вся бізнес-логіка реалізована з використанням стандартів фреймворку.

-Відмовостійкість

У разі збою оплати або відсутності доступу до сервера користувач повинен отримувати повідомлення про помилку та мати можливість повторного виконання операції.

-Конфіденційність даних

Персональні та платіжні дані мають зберігатись згідно з нормами GDPR (у разі масштабування на ЄС), з доступом лише для авторизованих користувачів і адміністрації.

1.4 Висновки до розділу 1

У розділі проведено комплексний аналіз сучасних технологій підбору комп’ютерних комплектуючих та визначено ключові принципи формування оптимальної апаратної конфігурації персонального комп’ютера. Встановлено, що індивідуальна збірка ПК є більш ефективним підходом, ніж купівля готових рішень, оскільки забезпечує повний контроль над якістю елементної бази, адаптацію до специфіки цільового використання, зменшення непотрібних витрат та можливість технологічної оптимізації системи. Доведено, що самостійний підбір компонентів сприяє не лише підвищенню продуктивності та надійності, а й розвитку технічної компетентності користувача.

Під час дослідження встановлено, що основними критеріями якісного складання є сумісність компонентів, енергетичний баланс, тепловий режим, оптимальний бюджет та відсутність “вузьких місць” у системі. Підкреслено, що

неправильний підбір навіть одного елементу може призвести до критичної несумісності, що зумовлює фінансові втрати та зниження ефективності.

Особливу увагу приділено значенню сучасних цифрових інструментів підбору, які дозволяють автоматизувати процес формування системи та мінімізувати ризик помилок. Розглянуті конфігуратори TELEMART та COMPX підтвердили доцільність використання онлайн-сервісів при складанні ПК — завдяки інтегрованим механізмам перевірки сумісності, наявності рекомендацій та можливості адаптації конфігурації під конкретні задачі або оновлення існуючої системи.

Результати аналізу демонструють, що структурна збалансованість конфігурації є ключовим фактором надійності та продуктивності. Надмірна концентрація на одному компоненті (наприклад, на процесорі або GPU) без відповідного підтримуючого апаратного середовища призводить до часткових втрат потенціалу системи. Найбільш критичним є узгодження процесора та відеокарти, особливо у високопродуктивних та ігрових конфігураціях. Здійснено постановку задачі на майбутню розробку.

2 МОДЕЛЮВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ З ПІДБОРУ КОМП'ЮТЕРНИХ КОМПЛЕКТУЮЧИХ

2.1 Проектування діаграми класів автоматизованої системи з підбору комп'ютерних комплектуючих

Діаграми класів в рамках уніфікованої мови моделювання UML виступають ключовим засобом структурного опису програмних систем на концептуальному, логічному та фізичному рівнях проєктування. Вони належать до інструментів статичного моделювання, адже відображають не послідовність виконання процесів, а архітектоніку системи, включаючи її об'єкти, їхні властивості, взаємозв'язки та принципи організації. Основне призначення діаграм класів полягає у формуванні формалізованої, чітко визначеної та графічно структурованої моделі, яка демонструє склад системи та характер взаємодії між її елементами.

Таке моделювання забезпечує побудову логічної структури програмного забезпечення, що відображає категорії об'єктів, їхні атрибутивні характеристики та методи. Діаграма відображає механізми обробки, передачі та збереження даних, а також визначає ієрархічні співвідношення, включно з принципами наслідування, агрегації та композиції. Завдяки цьому система набуває структурної визначеності, що дозволяє чітко розмежувати функціональність модулів — від керування логікою до взаємодії з користувачем або здійснення обчислень.

Застосування діаграм класів на ранніх етапах проєктування дає змогу формалізувати концепцію програмного продукту, забезпечити єдине розуміння архітектури серед членів команди та знизити ймовірність логічних невідповідностей на стадії реалізації. Вони сприяють модульності, масштабованості та підвищенню ефективності повторного використання

компонентів. Як складова технічної документації, діаграми класів фіксують архітектурні рішення, забезпечують трасування зв’язків між елементами та виступають опорою під час аналізу потенційних змін.

В межах життєвого циклу програмного забезпечення такі діаграми використовуються не лише під час проєктування, але й у процесі супроводу та модернізації системи. Вони допомагають підтримувати відповідність між початковою концептуальною моделлю та реалізованим програмним кодом, полегшують введення нових спеціалістів у проєкт і забезпечують ефективну підтримку та розвиток програмної системи. На рисунку 2.1 наведемо діаграму класів автоматизованої системи з підбору комп’ютерних комплектуючих.

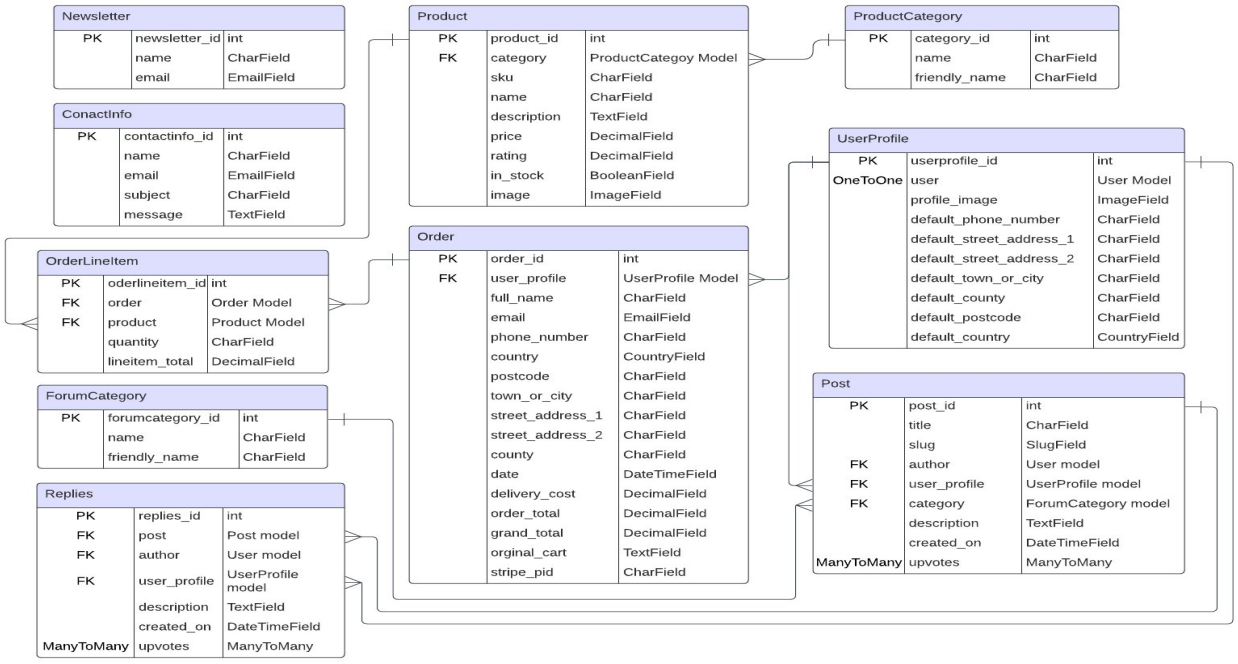


Рисунок 2.1 – Діаграма класів автоматизованої системи з підбору комп’ютерних комплектуючих

Клас Product є центральною сутністю системи та описує інформаційні одиниці комп’ютерних комплектуючих. Він містить ідентифікаційний атрибут product_id (PK), що забезпечує унікальність записів, а також атрибути sku, name,

description, price, rating, in_stock, image. Структурно цей клас представляє логічну модель апаратних компонентів, доступних для підбору чи придбання. Його зв'язок FK → ProductCategory реалізує класифікацію, дозволяє здійснювати фільтрацію та формує ієрархію типів комплектуючих, що є основою для подальшого аналізу сумісності.

Клас ProductCategory виконує роль таксономічної моделі, забезпечуючи логічне групування товарів. Атрибути category_id (PK), name та friendly_name дозволяють системі застосовувати частково формалізовані, частково орієнтовані на користувача назви. Взаємозв'язок з Product реалізує тип відношення "один до багатьох", визначаючи категоріальну приналежність кожного компонента (наприклад, процесор, пам'ять, графічна карта).

Клас Order моделює транзакційну структуру оформлення підбору чи покупки. order_id є первинним ключем; атрибут user_profile як зовнішній ключ пов'язує замовлення з конкретним користувачем. Поля full_name, email, phone_number, country, postcode, town_or_city, street_address_1, street_address_2, county забезпечують збереження параметрів доставки та ідентифікації замовника. delivery_cost, order_total, grand_total відображають економічну складову. date фіксує момент здійснення операції, а stripe_pid – унікальний ідентифікатор платіжної транзакції. Таким чином, Order виконує роль інтегратора результатів конфігураційного підбору, підтверджуючи коректність і фіналізацію ухваленого рішення.

Клас OrderLineItem деталізує складові замовлення. orderlineitem_id (PK) у поєднанні з order та product (FK) формує логічну структуру "позиція у замовленні", чим дозволяє аналізувати, які саме комплектуючі були обрані та як вони поєднуються між собою. quantity та lineitem_total використовуються при підрахунку вартості та формуванні рекомендацій на основі частоти вибору компонентів у поєднаннях.

Клас UserProfile моделює користувацький профіль. userprofile_id (PK) поєднується через зв’язок OneToOne з базовим обліковим записом користувача (User model). Він містить атрибути profile_image, default_phone_number, default_street_address_1, default_street_address_2, default_town_or_city, default_county, default_postcode, default_country. Цей клас реалізує персоналізацію процесу підбору, дозволяє проводити повторні конфігурації з урахуванням попереднього вибору та аналізу вподобань.

Клас Post представлений атрибутами post_id (PK), title, slug, description, created_on. Зовнішні ключі author (User model), user_profile (UserProfile model) та category (ForumCategory model) забезпечують джерело дискусійної інформації та інтеграцію експертного досвіду користувачів. Наявність зв’язку ManyToMany (upvotes) формує систему соціального підтвердження корисності інформації, яка у перспективі може використовуватись для алгоритмічних рекомендацій.

Клас Replies деталізує взаємодію в межах форуму. replies_id (PK) пов’язується з post, author (User model) та user_profile, що створює картину комунікацій між учасниками системи. description, created_on, upvotes підтримують механізм зворотного зв’язку та соціальної оцінки рішень. Цей елемент є ключовим для еволюції бази знань щодо підбору комплектуючих.

Клас ForumCategory структурує інформацію в межах обговорень. forumcategory_id (PK), name та friendly_name дозволяють формувати тематизоване середовище, де користувачі отримують експертні уточнення щодо конфігурацій, продуктивності та сумісності компонентів.

Клас ContactInfo виконує роль інтерфейсу зв’язку з технічною або консультаційною підтримкою. contactinfo_id (PK), name, email, subject, message дозволяють фіксувати запити щодо складних технічних сценаріїв.

Клас Newsletter реалізує механізм інформаційної підтримки. newsletter_id (PK), name та email використовуються для надіслання рекомендацій, аналітики

про оновлення продукції, зміну цін, появу нових компонентів, а також попередження про сумісність між моделями.

2.2 Проектування діаграми послідовності автоматизованої системи з підбору комп'ютерних комплектуючих

Діаграма послідовності є одним із фундаментальних типів UML-діаграм, призначених для моделювання динамічної поведінки програмних систем. Вона відображає взаємодію об'єктів протягом часу, демонструючи, як об'єкти обмінюються повідомленнями для виконання конкретних бізнес-процесів або функцій системи. Такий підхід дозволяє детально дослідити логіку роботи системи, визначити оптимальну послідовність операцій та виявити можливі проблеми у взаємодії об'єктів.

На діаграмі кожен об'єкт представлений вертикальною лінією життя, що відображає його активність у часі. Горизонтальні стрілки між цими лініями символізують передачу повідомлень або виклики методів. Послідовність цих звернень дає змогу відстежити логіку виконання сценарію від початкового запиту до остаточного результату. Діаграма також враховує реакції об'єктів на отримані повідомлення, включаючи можливі відповіді або повернені значення.

Даний вид діаграм широко застосовується для моделювання варіантів використання системи, проектування бізнес-процесів та перевірки коректності функціональної реалізації. Він дозволяє чітко визначити, які компоненти взаємодіють між собою, які запити надсилаються та у якій логічній послідовності відбувається обмін інформацією. Завдяки часовій орієнтації можна розрізняти синхронні та асинхронні виклики, а також враховувати умови виконання, цикли або паралельні дії.

Практичне використання діаграм послідовності забезпечує докладне описання процесів на етапах аналізу та проектування, підвищуючи узгодженість функцій та точність їх реалізації. Особливо це актуально для складних систем із багатокроковими взаємодіями, де важлива координація між компонентами. Завдяки цьому підходу зменшується ймовірність логічних помилок, підвищується прозорість взаємодій і полегшується комунікація між аналітиками, розробниками, тестувальниками та іншими учасниками проєкту.

На рисунку 2.2 наведемо діаграму послідовності автоматизованої системи з підбору комп’ютерних комплектуючих.

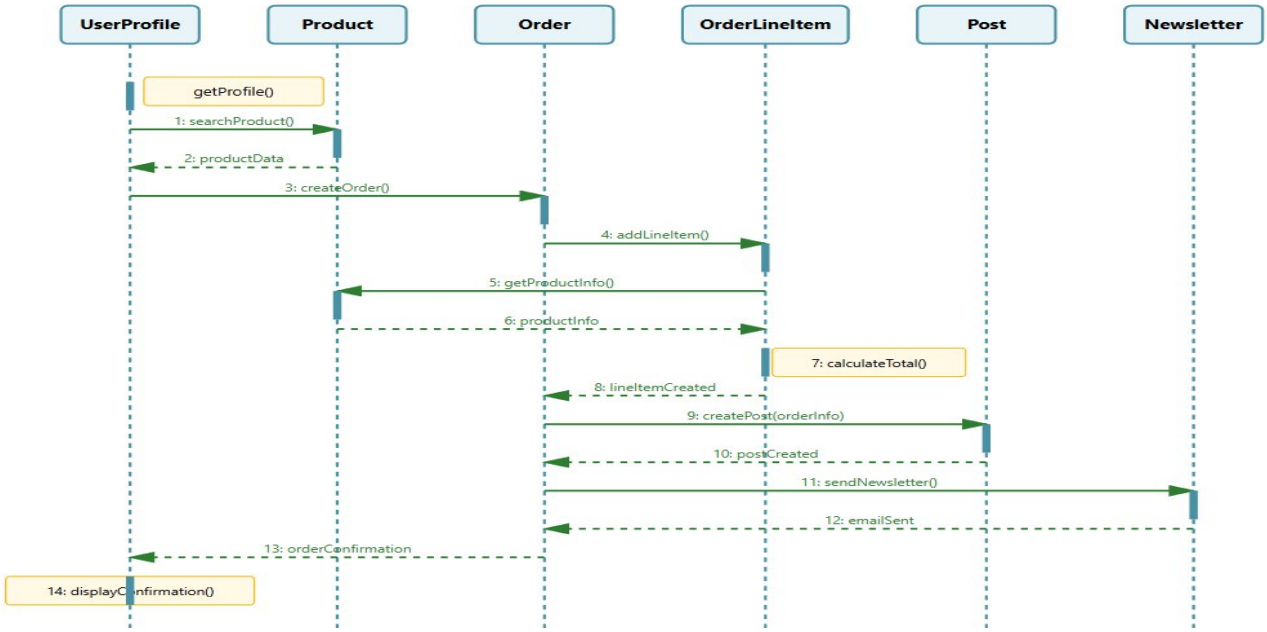


Рисунок 2.2 – Діаграма послідовності автоматизованої системи з підбору комп’ютерних комплектуючих

Діаграма послідовності ілюструє процес оформлення замовлення користувачем в системі електронної комерції, демонструючи взаємодію між п'ятьма основними компонентами системи: UserProfile, Product, Order, OrderLineItem, Post та Newsletter.

Процес починається з того, що об'єкт UserProfile ініціює операцію перегляду власного профілю, викликаючи метод getProfile, який є внутрішнім методом для отримання даних поточного користувача. Після успішного отримання інформації про профіль користувач переходить до наступного етапу взаємодії із системою.

На другому етапі UserProfile відправляє синхронний запит до об'єкта Product, викликаючи метод searchProduct. Цей виклик передає управління компоненту Product, який активується та починає обробку запиту на пошук товару. Компонент Product виконує внутрішню логіку пошуку, яка може включати звернення до бази даних, фільтрацію товарів за певними критеріями та формування результатів пошуку. Після завершення обробки запиту Product повертає дані про знайдені товари назад до UserProfile через повернення productData. Це повернення відбувається асинхронно, що позначено пунктирною лінією на діаграмі.

Отримавши дані про продукти, користувач приймає рішення про покупку і UserProfile ініціює третій крок процесу, відправляючи синхронний запит createOrder до компонента Order. Цей виклик передає всю необхідну інформацію для створення нового замовлення, включаючи дані користувача, обраний товар та інші параметри замовлення. Компонент Order активується та починає процес формування замовлення, виділяючи для цього необхідні ресурси системи та ініціалізуючи внутрішні структури даних.

На четвертому етапі компонент Order делегує частину роботи компоненту OrderLineItem, викликаючи метод addLineItem. Цей виклик необхідний для додавання конкретної позиції товару до замовлення. OrderLineItem є окремою сутністю, яка представляє один рядок у замовленні і містить інформацію про конкретний товар, його кількість, ціну та інші атрибути. Компонент OrderLineItem активується і починає формування позиції замовлення.

Для коректного формування позиції замовлення OrderLineItem потребує детальної інформації про продукт, тому на п'ятому кроці він відправляє синхронний запит getProductInfo назад до компонента Product. Цей зворотний виклик демонструє, що взаємодія в системі не завжди є лінійною і компоненти можуть звертатися один до одного в різних напрямках залежно від потреб бізнес-логіки. Product активується повторно, обробляє запит та збирає всю необхідну інформацію про товар, включаючи його назву, опис, ціну, наявність на складі та інші характеристики.

На шостому кроці Product повертає зібрану інформацію назад до OrderLineItem через асинхронне повернення productInfo. OrderLineItem отримує всі необхідні дані та може продовжити формування позиції замовлення з повною інформацією про продукт.

Сьомий крок є внутрішньою операцією компонента OrderLineItem, де викликається метод calculateTotal. Цей метод відповідає за розрахунок загальної вартості позиції замовлення, враховуючи ціну товару, кількість, можливі знижки, податки та інші фінансові параметри. Це критично важлива операція, яка забезпечує коректність фінансових розрахунків у системі. OrderLineItem виконує всі необхідні обчислення і зберігає результат у своїй внутрішній структурі даних.

Після завершення всіх операцій з формування позиції замовлення на восьмому кроці OrderLineItem повертає підтвердження назад до компонента Order через асинхронне повернення lineItemCreated. Це повідомлення сигналізує про успішне створення позиції замовлення і дозволяє Order продовжити обробку замовлення в цілому.

На дев'ятому етапі компонент Order ініціює створення публікації про замовлення, відправляючи синхронний запит createPost до компонента Post і передаючи параметр orderInfo, який містить інформацію про замовлення. Компонент Post відповідає за створення публічних записів або повідомлень про

замовлення, які можуть бути видимі іншим користувачам системи або використовуватися для внутрішніх цілей відстеження історії замовлень. Post активується і створює відповідний запис у системі.

Десятий крок представляє асинхронне повернення postCreated від компонента Post назад до Order, що підтверджує успішне створення публікації про замовлення. Це дозволяє Order відслідковувати, що всі пов'язані дії з замовленням виконані коректно.

На одинадцятому етапі Order ініціює відправку email розсилки користувачу, викликаючи метод sendNewsletter у компоненті Newsletter. Цей виклик передає всю необхідну інформацію для формування та відправки електронного листа з підтвердженням замовлення, деталями покупки та іншою релевантною інформацією. Компонент Newsletter активується і починає процес формування та відправки email повідомлення через зовнішні сервіси електронної пошти.

Дванадцятий крок представляє асинхронне повернення emailSent від Newsletter до Order, що підтверджує успішну відправку email користувачу. Це важливе підтвердження, оскільки воно засвідчує, що користувач буде поінформований про своє замовлення через додатковий канал комунікації.

На тринадцятому етапі компонент Order завершує весь процес обробки замовлення і повертає orderConfirmation назад до UserProfile через асинхронне повернення. Це підтвердження містить всю інформацію про успішно створене замовлення, включаючи номер замовлення, деталі позицій, загальну суму та інші важливі дані.

Нарешті, на чотирнадцятому і останньому кроці UserProfile викликає внутрішній метод displayConfirmation, який відповідає за відображення підтвердження замовлення користувачеві в інтерфейсі системи. Цей метод формує візуальне представлення підтвердження і показує його користувачеві, завершуючи таким чином весь процес оформлення замовлення.

2.3 Проектування структурної схеми автоматизованої системи з підбору комп’ютерних комплектуючих

Структурна схема системи електронної комерції являє собою багаторівневу архітектуру, яка організована за принципом розділення відповідальностей та забезпечує високу масштабованість, надійність та підтримуваність всієї системи. Архітектура побудована на основі чотирьох основних рівнів, кожен з яких виконує специфічні функції та взаємодіє з іншими рівнями через чітко визначені інтерфейси (рис.2.3).

Верхній рівень архітектури представлений рівнем презентації, який є точкою входу для всіх користувачів системи та забезпечує візуальну взаємодію між користувачами та функціональністю системи. Цей рівень включає п'ять основних компонентів, кожен з яких орієнтований на різні типи користувачів та сценарії використання. Веб-інтерфейс є основним способом доступу до системи через браузер і побудований з використанням сучасних веб-технологій, включаючи HTML для структури сторінок, CSS для стилізації та оформлення, JavaScript для інтерактивності та динамічної поведінки. Цей інтерфейс реалізує адаптивний дизайн, що дозволяє коректно відображати контент на екранах різних розмірів та пристроях, від великих настільних моніторів до планшетів та смартфонів. Веб-інтерфейс також включає персональну панель користувача, де кожен клієнт може управляти своїм профілем, переглядати історію замовлень, відстежувати статус доставки та налаштовувати персональні параметри системи.

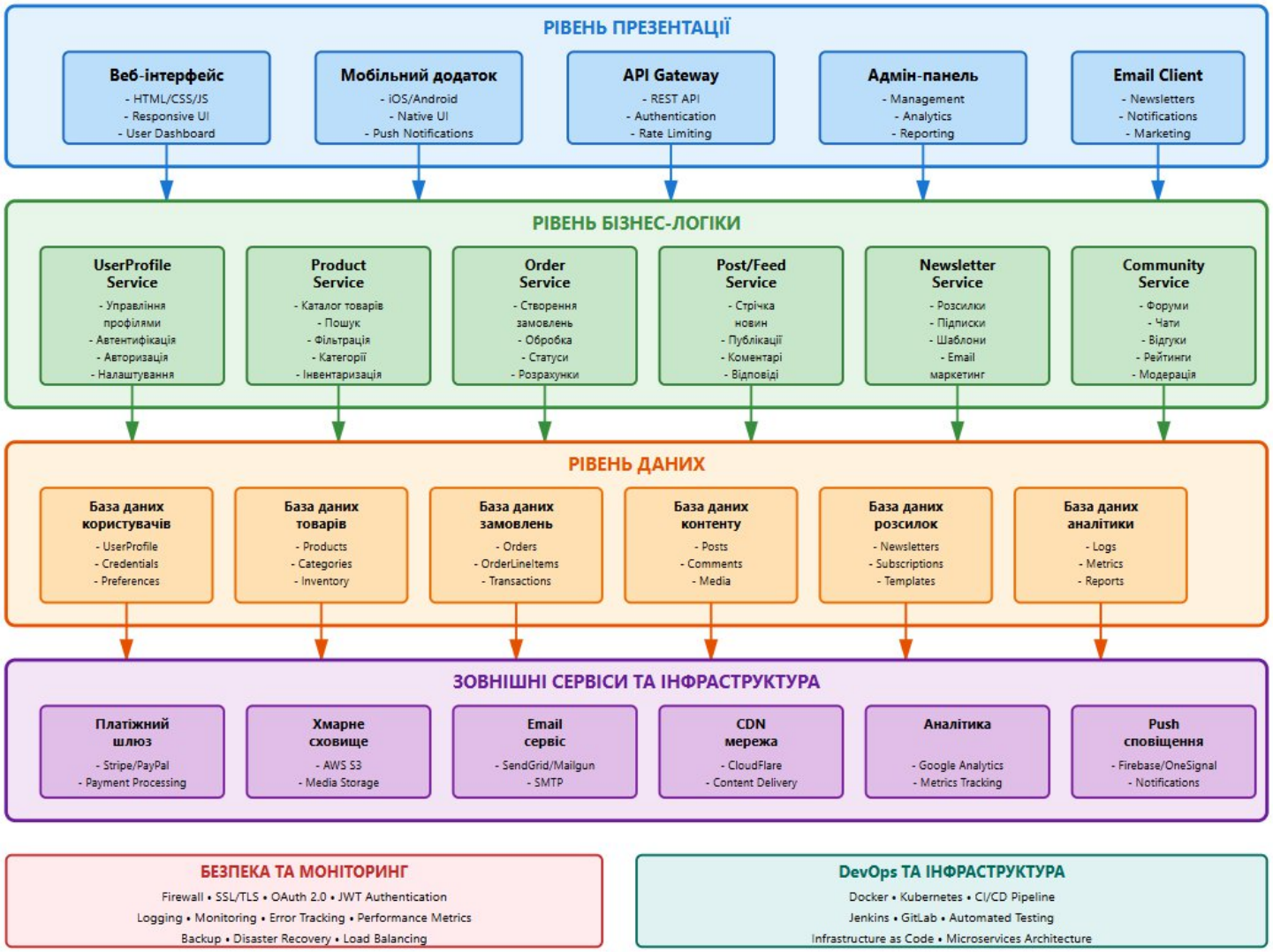


Рисунок 2.3 – Структурна схема автоматизованої системи з підбору комп’ютерних комплектуючих

Поруч з веб-інтерфейсом функціонує мобільний додаток, який забезпечує нативний досвід використання системи на мобільних платформах iOS та Android. Мобільний додаток розроблений з урахуванням специфіки мобільних пристроїв та надає повний набір функціональності, доступної через веб-інтерфейс, але оптимізований для сенсорного управління та роботи на менших екранах. Додаток реалізує нативний користувацький інтерфейс для кожної платформи, забезпечуючи природну взаємодію та відповідність гайдлайнам Apple Human Interface Guidelines для iOS та Material Design для Android. Особливою перевагою мобільного додатку є підтримка push-сповіщень, які дозволяють системі оперативно інформувати користувачів про важливі події, такі як зміна статусу замовлення, спеціальні пропозиції, нові повідомлення чи відповіді на коментарі.

API Gateway є критично важливим компонентом рівня презентації, який виступає центральною точкою входу для всіх зовнішніх запитів до системи. Gateway реалізує REST архітектурний стиль, забезпечуючи стандартизований та зрозумілий інтерфейс для взаємодії клієнтських додатків з серверною частиною системи. Цей компонент відповідає за автентифікацію користувачів, перевірку їхніх облікових даних та видачу токенів доступу, які використовуються для подальших запитів до системи. API Gateway також реалізує механізми обмеження швидкості запитів, що захищає систему від зловживань та атак типу відмови в обслуговуванні, дозволяючи контролювати кількість запитів від кожного клієнта в одиницю часу. Крім того, Gateway виконує маршрутизацію запитів до відповідних сервісів бізнес-логіки, агрегацію відповідей від кількох сервісів, трансформацію даних та обробку помилок.

Адміністративна панель призначена для персоналу компанії, який управляє системою та здійснює моніторинг її роботи. Через цей інтерфейс адміністратори можуть виконувати широкий спектр управлінських операцій,

включаючи додавання нових товарів до каталогу, оновлення інформації про існуючі продукти, управління категоріями товарів, модерацію користувацького контенту, обробку проблемних замовлень та управління правами доступу користувачів. Панель також надає потужні аналітичні інструменти, які дозволяють відстежувати ключові метрики бізнесу, такі як обсяг продажів, конверсія, середній чек, найпопулярніші товари, поведінку користувачів та ефективність маркетингових кампаній. Система звітності дозволяє генерувати детальні звіти за різними періодами часу та експортувати їх у різні формати для подальшого аналізу.

Email клієнт завершує рівень презентації і відповідає за всі форми email комунікації з користувачами системи. Цей компонент забезпечує відправку різноманітних типів повідомлень, включаючи регулярні розсилки новин про нові товари та спеціальні пропозиції, транзакційні сповіщення про підтвердження замовлень та зміни їхнього статусу, нагадування про покинуті кошики покупок, запрошення до участі в опитуваннях та програмах лояльності, а також таргетовані маркетингові кампанії на основі поведінки та переваг користувачів.

Другий рівень системи представлений рівнем бізнес-логіки, який є серцем всієї архітектури та містить основну функціональність системи. Цей рівень організований за принципами сервіс-орієнтованої архітектури та складається з шести незалежних сервісів, кожен з яких відповідає за конкретну бізнес-область. UserProfile Service є відповідальним за всі аспекти управління обліковими записами користувачів та їхніми профілями. Цей сервіс обробляє процеси реєстрації нових користувачів, включаючи валідацію даних, перевірку унікальності email адреси та створення облікового запису. Сервіс реалізує механізми автентифікації, підтримуючи різні методи входу в систему, включаючи класичну комбінацію email та пароля, а також соціальну автентифікацію через облікові записи Google, Facebook або інші провайдери.

Компонент авторизації перевіряє права доступу користувачів до різних функцій системи на основі їхніх ролей та дозволів. UserProfile Service також управляє персональними налаштуваннями користувачів, включаючи мовні преференції, налаштування конфіденційності, параметри сповіщень, збережені адреси доставки та платіжну інформацію.

Product Service відповідає за управління всім життєвим циклом товарів в системі. Цей сервіс підтримує повний каталог товарів з детальною інформацією про кожен продукт, включаючи назву, опис, технічні характеристики, ціну, наявність на складі, зображення та відгуки покупців. Сервіс реалізує потужний пошуковий механізм, який дозволяє користувачам швидко знаходити потрібні товари за ключовими словами, артикулами або іншими параметрами. Система фільтрації надає можливість звужувати результати пошуку за множинними критеріями, такими як ціновий діапазон, бренд, характеристики, рейтинг, наявність на складі та багато інших параметрів залежно від категорії товару. Product Service управляє ієрархічною структурою категорій та підкатегорій товарів, що полегшує навігацію по каталогу та організацію продуктів. Компонент інвентаризації відстежує кількість товарів на складах, автоматично оновлює статус наявності, резервує товари під час оформлення замовлення та інтегрується з системами управління складом для синхронізації даних про запаси.

Order Service є ключовим компонентом для обробки всіх транзакцій в системі. Цей сервіс керує повним життєвим циклом замовлення від моменту його створення до фінальної доставки або скасування. Процес створення замовлення включає валідацію вибраних товарів, перевірку їхньої наявності на складі, резервування необхідної кількості, розрахунок вартості з урахуванням знижок та промокодів, обчислення вартості доставки на основі адреси та обраного методу, розрахунок податків згідно з чинним законодавством та створення запису замовлення в базі даних. Сервіс обробки замовлень координує

всі етапи виконання, включаючи підтвердження оплати, передачу інформації на склад для збирання, організацію доставки через інтеграцію з логістичними партнерами та комунікацію з клієнтом щодо статусу замовлення. Система управління статусами відстежує поточний стан кожного замовлення через визначені стани, такі як створено, оплачено, збирається на складі, відправлено, в дорозі, доставлено або скасовано. Компонент розрахунків забезпечує точні фінансові операції, включаючи обчислення знижок, застосування промокодів, розрахунок податків, обробку повернень та відшкодувань.

Post Service або Feed Service відповідає за створення та управління контентом соціальної складової платформи. Цей сервіс підтримує стрічку новин, де користувачі можуть бачити оновлення від інших покупців, рекомендації товарів, огляди та статті від експертів, анонси нових надходжень та спеціальних пропозицій. Користувачі можуть створювати власні публікації, ділитися враженнями від покупок, завантажувати фотографії товарів в використанні, ставити запитання спільноті та надавати поради іншим покупцям. Система коментарів дозволяє вести дискусії під публікаціями, створюючи інтерактивне середовище для обміну думками. Компонент відповідей забезпечує можливість реагувати на коментарі, створювати вкладені обговорення та згадувати інших користувачів для залучення їх до дискусії. Post Service також реалізує механізми модерації контенту для забезпечення дотримання правил спільноти та видалення неприйнятного вмісту.

Newsletter Service спеціалізується на управлінні всіма аспектами email маркетингу та комунікації з користувачами. Цей сервіс керує базою підписників, дозволяючи користувачам оформляти підписку на різні типи розсилок відповідно до їхніх інтересів, такі як щотижневі дайджести нових товарів, спеціальні пропозиції та розпродажі, рекомендації на основі історії покупок, новини компанії та галузі, освітній контент про продукти. Система підтримує бібліотеку шаблонів email повідомлень з різним дизайном та

структурою для різних типів комунікації. Newsletter Service реалізує сегментацію аудиторії, дозволяючи відправляти таргетовані повідомлення різним групам користувачів на основі їхніх характеристик, поведінки, історії покупок та інших параметрів. Компонент email маркетингу включає інструменти для A/B тестування різних версій листів, аналізу ефективності кампаній через метрики відкриттів, кліків та конверсій, автоматизації послідовностей листів та персоналізації контенту для кожного отримувача.

Community Service завершує рівень бізнес-логіки і відповідає за функціональність спільноти користувачів та взаємодію між ними. Цей сервіс підтримує форуми, де користувачі можуть створювати теми для обговорення, ставити запитання про товари та отримувати відповіді від інших покупців або представників компанії. Компонент чатів забезпечує можливість миттєвого обміну повідомленнями між користувачами, включаючи приватні бесіди та групові чати. Система відгуків дозволяє користувачам залишати детальні огляди куплених товарів, включаючи текстовий опис досвіду використання, оцінку за різними критеріями та завантаження фотографій або відео. Механізм рейтингів агрегує оцінки від користувачів для формування загального рейтингу товарів та надійності продавців. Інструменти модерації надають можливість відстежувати активність в спільноті, виявляти та видаляти спам, неприйнятний контент або порушення правил, а також управляти користувачами, які систематично порушують правила спільноти.

Третій рівень архітектури представлений рівнем даних, який забезпечує постійне зберігання всієї інформації системи та ефективний доступ до неї. Цей рівень організований за принципом розділення даних на спеціалізовані бази даних відповідно до їхнього призначення та характеристик використання. База даних користувачів зберігає всю інформацію про зареєстрованих користувачів системи, включаючи профілі з особистою інформацією, контактними даними, адресами доставки та платіжною інформацією. Облікові дані містять хешовані

паролі, токени сесій, історію входів в систему та параметри безпеки облікового запису. Персональні налаштування користувачів зберігають їхні преференції щодо інтерфейсу, мови, сповіщень, приватності та інших параметрів персоналізації досвіду використання системи.

База даних товарів є центральним сховищем інформації про всі продукти, доступні в системі. Вона містить повні записи продуктів з усіма атрибутами, включаючи назви на різних мовах, детальні описи, технічні специфікації, ціни в різних валютах, посилання на зображення та медіафайли, SEO метадані та історію змін. Ієрархічна структура категорій та підкатегорій організує продукти в логічні групи, полегшуючи навігацію та пошук. Дані інвентаризації відстежують поточну кількість кожного товару на різних складах, зарезервовану кількість під активні замовлення, історію руху товарів та прогнози потреби в поповненні запасів. База також зберігає відносини між продуктами, такі як аксесуари, супутні товари, альтернативи та комплекти.

База даних замовлень містить повну інформацію про всі транзакції в системі. Основні записи замовлень включають інформацію про замовника, дату створення, поточний статус, адресу доставки, обраний метод доставки, платіжну інформацію та загальну вартість. Кожне замовлення пов'язане з набором позицій через таблицю OrderLineItems, яка зберігає інформацію про конкретні товари в замовленні, їхню кількість, ціну на момент покупки та застосовані знижки. Фінансові транзакції відстежують всі платіжні операції, включаючи авторизації, списання коштів, повернення та відшкодування. База також зберігає історію змін статусів замовлень, комунікації з клієнтами та примітки для внутрішнього використання.

База даних контенту зберігає весь користувацький та редакційний контент соціальної частини платформи. Публікації містять текстовий вміст, метадані автора, дату створення, категорію та статус модерації. Коментарі організовані в ієрархічні структури, що дозволяє створювати вкладені обговорення будь-якої

глибини. Медіафайли включають зображення, відео та інші типи файлів, завантажені користувачами, з метаданими про розміри, формати та права доступу. База також зберігає інформацію про лайки, репости, збереження публікацій та інші форми взаємодії користувачів з контентом.

База даних розсилок управляє всією інформацією, пов'язаною з email маркетингом. Записи розсилок містять шаблони листів, контент для відправки, налаштування сегментації аудиторії та параметри планування відправки. Підписки відстежують, які користувачі підписані на які типи розсилок, коли вони підписалися та чи активна підписка. Шаблони email повідомлень зберігаються як в HTML форматі для сучасних поштових клієнтів, так і в текстовому форматі для базової підтримки. База також містить статистику відправлених кампаній, включаючи кількість доставлених листів, відкриттів, кліків та відписок.

База даних аналітики зберігає величезні обсяги даних про функціонування системи та поведінку користувачів. Логи системи записують всі значущі події, помилки, попередження та інформаційні повідомлення з різних компонентів системи для подальшого аналізу та діагностики проблем. Метрики збирають кількісні показники роботи системи, такі як час відповіді різних ендпоінтів, використання ресурсів серверів, кількість активних користувачів, обсяг трафіку та багато інших параметрів. Звіти містять агреговані дані за різні періоди часу, що використовуються для бізнес-аналітики, виявлення трендів та прийняття стратегічних рішень.

Четвертий та найнижчий рівень архітектури представлений зовнішніми сервісами та інфраструктурою, які надають критично важливі функції, що не мають сенсу розробляти власноруч з огляду на складність, вартість або необхідність спеціалізованої експертизи. Платіжний шлюз є інтеграцією з професійними провайдерами платіжних послуг, такими як Stripe або PayPal, які забезпечують безпечну обробку платіжних карток та інших методів оплати. Ці

сервіси беруть на себе складність дотримання стандартів безпеки платіжних даних PCI DSS, обробки різних типів карток та платіжних методів, виявлення та попередження шахрайських транзакцій, обробки повернень та спорів, конвертації валют та багатовалютних розрахунків. Інтеграція з платіжним шлюзом дозволяє системі приймати платежі без необхідності зберігати чутливі платіжні дані користувачів, значно зменшуючи ризики безпеки та регуляторні вимоги.

Хмарне сховище, зазвичай реалізоване на базі Amazon S3 або аналогічних сервісів, забезпечує надійне та масштабоване зберігання медіафайлів, таких як зображення товарів, фотографії від користувачів, відеоогляди, документація та інші типи файлів. Хмарне сховище надає переваги необмеженої масштабованості, високої доступності та географічного розподілу даних, автоматичного резервного копіювання та версіонування файлів, економічної ефективності завдяки гнучким тарифам оплати за фактичне використання, інтеграції з CDN мережами для швидкої доставки контенту користувачам по всьому світу. Система може зберігати кілька версій кожного зображення в різних розмірах та форматах для оптимізації завантаження на різних пристроях.

Email сервіс представлений інтеграцією з професійними провайдерами email доставки, такими як SendGrid або Mailgun, які спеціалізуються на масовій та транзакційній email розсилці. Ці сервіси забезпечують високий рівень доставки листів завдяки підтримці репутації IP адрес та доменів, управлінню відписками та скаргами на спам відповідно до законодавчих вимог, детальну аналітику доставки, відкриттів та кліків, обробку відмов та помилок доставки, підтримку різних типів автентифікації email, таких як SPF, DKIM та DMARC, шаблонізацію та персоналізацію листів, A/B тестування різних версій повідомлень. Використання спеціалізованого email сервісу значно підвищує імовірність того, що листи потраплять до папки вхідних, а не будуть заблоковані як спам.

CDN мережа або мережа доставки контенту, зазвичай реалізована на базі CloudFlare або аналогічних сервісів, забезпечує швидку доставку статичного контенту користувачам незалежно від їхнього географічного розташування. CDN кешує копії статичних файлів, таких як зображення, стилі CSS, JavaScript файли та інші ресурси на серверах, розташованих по всьому світу, дозволяючи користувачам завантажувати їх з найближчого сервера. Це значно зменшує час завантаження сторінок та покращує загальний досвід користування. CDN також надає додаткові переваги, такі як захист від DDoS атак, SSL сертифікати для безпечного з'єднання, стиснення контенту для зменшення обсягу переданих даних, оптимізація зображень на льоту залежно від пристрою користувача.

Сервіс аналітики, представлений інтеграцією з Google Analytics або подібними системами, забезпечує глибоке розуміння поведінки користувачів на сайті та в додатку. Цей сервіс відстежує велику кількість метрик, включаючи відвідуваність сторінок, джерела трафіку, демографічні дані аудиторії, шляхи користувачів по сайту, конверсійні воронки, відмови та багато іншого. Аналітичні дані використовуються для оптимізації користувацького досвіду, виявлення проблемних місць в інтерфейсі, оцінки ефективності маркетингових кампаній, прийняття рішень про розвиток продукту. Інтеграція з аналітичними системами дозволяє створювати детальні звіти, налаштовувати цілі та події для відстеження ключових дій користувачів.

Сервіс push-сповіщень, реалізований через Firebase Cloud Messaging або OneSignal, забезпечує можливість відправляти миттєві сповіщення на мобільні пристрої та браузери користувачів навіть коли додаток не активний. Цей канал комунікації використовується для інформування про важливі події, такі як зміна статусу замовлення, відповіді на коментарі, спеціальні пропозиції, що закінчуються, нагадування про покинуті кошики, персоналізовані рекомендації. Push-сповіщення мають високий рівень видимості та залученості порівняно з email, оскільки відображаються безпосередньо на екрані пристрою користувача.

Сервіс підтримує сегментацію аудиторії, планування відправки, A/B тестування різних повідомлень та детальну аналітику ефективності.

Окрім чотирьох основних рівнів, архітектура включає два горизонтальних компоненти, які охоплюють всю систему та забезпечують критично важливі наскрізні функції. Компонент безпеки та моніторингу забезпечує захист системи від різноманітних загроз та постійний нагляд за її станом. Firewall фільтрує вхідний та вихідний трафік, блокуючи підозрілі з'єднання та атаки. Шифрування SSL/TLS захищає всі дані, що передаються між клієнтами та сервером, запобігаючи їх перехопленню. Протокол OAuth 2.0 забезпечує безпечну авторизацію додатків третіх сторін без розкриття паролів користувачів. JWT токени використовуються для автентифікації користувачів та передачі інформації про їхні права доступу між різними компонентами системи. Система логування записує всі значущі події для подальшого аналізу та розслідування інцидентів. Моніторинг відстежує роботу всіх компонентів системи в реальному часі, виявляє аномалії та автоматично сповіщає адміністраторів про проблеми. Відстеження помилок збирає інформацію про виключення та збої в додатках для швидкого виправлення багів. Метрики продуктивності допомагають виявляти вузькі місця та оптимізувати роботу системи. Регулярне резервне копіювання забезпечує можливість відновлення даних у випадку катастрофічного збою. План відновлення після катастроф описує процедури швидкого відновлення роботи системи. Балансування навантаження розподіляє запити між кількома серверами для забезпечення високої доступності та продуктивності.

Компонент DevOps та інфраструктури забезпечує автоматизацію процесів розгортання, тестування та управління інфраструктурою. Контейнеризація за допомогою Docker дозволяє пакувати додатки разом з усіма залежностями в ізольовані контейнери, що гарантує однакову поведінку в різних середовищах. Оркестрація контейнерів через Kubernetes автоматизує розгортання,

масштабування та управління контейнеризованими додатками. Конвеєр безперервної інтеграції та доставки CI/CD автоматизує процес збирання, тестування та розгортання коду, дозволяючи швидко та безпечно випускати нові версії програмного забезпечення. Jenkins або GitLab CI виконують автоматичні перевірки коду, запускають тести та розгортають додатки в різні середовища. Автоматизоване тестування включає юніт-тести, інтеграційні тести, функціональні тести та тести продуктивності, що виконуються при кожній зміні коду. Інфраструктура як код дозволяє описувати та версіонувати конфігурацію серверів та мережі у вигляді коду, забезпечуючи відтворюваність та документування інфраструктури. Мікросервісна архітектура організує систему як набір невеликих незалежних сервісів, кожен з яких може розроблятися, тестуватися, розгортатися та масштабуватися незалежно від інших.

2.4 Висновки до розділу 2

У розділі 2 «Моделювання автоматизованої системи з підбору комп’ютерних комплектуючих» проведено комплексне моделювання архітектури та функціональних механізмів системи із застосуванням засобів UML, що охоплює як статичні, так і динамічні аспекти роботи програмного продукту, а також структурну організацію програмної платформи.У підрозділі 2.1 здійснено проєктування діаграми класів, яка виступає основним інструментом статичного моделювання та відображає логічну структуру системи. Визначено ключові класи: Product як центральна сутність для представлення комп’ютерних компонентів, ProductCategory для класифікації товарів, Order та OrderLineItem для управління транзакційними процесами оформлення замовлень, UserProfile для персоналізації та збереження даних

користувача, Post, Replies і ForumCategory для моделювання соціальної складової взаємодії користувачів, ContactInfo та Newsletter для комунікації та інформаційної підтримки. Кожен клас описаний із врахуванням атрибутів, методів і зв’язків, що забезпечує точну структуризацію даних та визначає механізми обробки і передачі інформації всередині системи. Побудована діаграма дозволяє формалізувати архітектурні рішення, забезпечити модульність, масштабованість та ефективність повторного використання компонентів, а також служить опорою для супроводу та модернізації системи.У підрозділі 2.2 розроблено діаграму послідовності, що демонструє динамічну взаємодію основних компонентів системи під час процесу оформлення замовлення користувачем. Процес охоплює чотирнадцять кроків, починаючи від перегляду профілю користувача і пошуку товару, до формування замовлення, додавання позицій, обробки інформації про продукт, розрахунку фінансових параметрів, створення публікацій у системі та відправки email-повідомлень. Діаграма послідовності дозволяє простежити логіку синхронних і асинхронних викликів між об’єктами, оцінити коректність та ефективність взаємодії, виявити потенційні вузькі місця, а також забезпечує прозорість роботи системи для аналітиків, розробників і тестувальників.У підрозділі 2.3 створено структурну схему багаторівневої архітектури системи, яка організована за принципом розділення відповідальностей і включає чотири рівні: презентації, бізнес-логіки, даних та інтеграційний. Рівень презентації включає веб-інтерфейс, мобільний додаток, API Gateway, адміністративну панель та email клієнт, що забезпечує повноцінну взаємодію користувачів із системою. Рівень бізнес-логіки реалізує сервіси UserProfile, Product, Order, Post, Newsletter та Community, які відповідають за управління профілями користувачів, каталогом товарів, обробку замовлень, створення контенту, email-комунікацію та підтримку спільноти.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ З ПІДБОРУ КОМП'ЮТЕРНИХ КОМПЛЕКТУЮЧИХ

3.1 Обґрунтування засобів розробки

Для розробки цього проекту було обрано комплекс технологій і сервісів, кожен з яких виконує специфічну функцію та забезпечує ефективність, масштабованість і зручність у підтримці веб-сайту. Нижче наведено детальний опис причин вибору кожного інструменту та його ролі в розробці:

HTML – використовується для створення структури веб-сторінок та базової розмітки контенту. Завдяки HTML можна організувати текст, зображення, посилання та інтерактивні елементи, створюючи основу для подальшого стилювання та логіки сайту.

CSS – застосовується для додавання індивідуальних стилів та оформлення макета сайту. CSS дозволяє налаштувати кольори, шрифти, розташування елементів і забезпечує адаптивність під різні розміри екранів, що підвищує користувацький досвід.

Python3 – основна мова програмування для реалізації бізнес-логіки проекту. Python забезпечує простоту у написанні коду, підтримку складних алгоритмів та інтеграцію з фреймворком Django, що дозволяє швидко розробляти функціональні веб-додатки.

JavaScript – використовується для додання інтерактивності на сторінках. У даному проекті застосовується для реалізації функцій таймауту повідомлень, що дозволяє динамічно управляти відображенням сповіщень користувачу без перезавантаження сторінки.

Django – обраний як основний веб-фреймворк для розробки проекту. Django забезпечує швидку та структуровану розробку, містить готові механізми для роботи з базами даних, маршрутизації URL, обробки форм, а також безпеки.

Використання Django дозволяє створювати надійні та масштабовані веб-додатки з мінімальними витратами часу.

Bootstrap v4.6.2 – використовується як фронтенд-фреймворк для швидкого та адаптивного створення інтерфейсу. Він надає готові стилі та компоненти, що спрощує дизайн, робить його сучасним і сумісним із різними браузерами та пристроями.

Font Awesome – дозволяє інтегрувати іконки у веб-сторінки. Іконки допомагають зробити інтерфейс більш зрозумілим та візуально привабливим, а також покращують UX за рахунок графічного підкріплення текстового контенту.

Jinja Templating з Django – використовується для рендерингу динамічного контенту всередині HTML. Завдяки цьому шаблони стають гнучкими, можна передавати дані з серверної частини у фронтенд і відображати їх у потрібному форматі, що робить сайт інтерактивним і персоналізованим.

Lucid Chart – застосовується для моделювання структури бази даних. Це дозволяє наочно відобразити зв’язки між таблицями, спрощує планування архітектури даних і допомагає уникнути помилок у проектуванні бази даних.

Crispy Forms – використовується для контролю відображення та поведінки Django форм. Цей пакет дозволяє легко налаштовувати валідацію, стилізацію та порядок полів форм, що покращує юзабіліті та робить форми більш привабливими.

ElephantSQL (PostgreSQL) – обрана реляційна база даних для зберігання інформації проекту. PostgreSQL забезпечує високу продуктивність, надійність, підтримку складних запитів і масштабованість, а ElephantSQL дозволяє зручно розгортати базу у хмарі як під час розробки, так і в продакшені.

Git – використовується для контролю версій коду. Git дозволяє відстежувати зміни, керувати гілками розробки та забезпечує безпечну роботу у команді, зменшуючи ризик втрати даних та конфліктів у коді.

GitPod – хмарне середовище розробки, яке забезпечує швидкий старт проєкту без необхідності локальної установки та конфігурації середовища.

Heroku – платформа для розгортання веб-додатків. Використання Heroku дозволяє швидко розгортати проект, автоматично обробляти залежності та забезпечує масштабованість без потреби власного сервера.

Gunicorn – WSGI-сервер для розгортання Python/Django додатків. Gunicorn відповідає за обробку HTTP-запитів, працює стабільно і ефективно, забезпечуючи швидкий обмін даними між сервером і додатком.

Allauth – бібліотека для системи аутентифікації користувачів. Вона підтримує реєстрацію, вхід через соціальні мережі, підтвердження email і відновлення паролю, що робить управління користувачами простим та безпечним.

AWS S3 – обраний сервіс для зберігання статичних та медіафайлів. S3 забезпечує надійне та масштабоване зберігання файлів, швидкий доступ та інтеграцію з Django.

AWS IAM – використовується для управління ролями та правами доступу до ресурсів AWS. Це дозволяє створювати безпечну систему доступу, обмежуючи дії користувачів і сервісів відповідно до ролей.

Stripe – сервіс для обробки платежів і вебхуків. Stripe забезпечує безпечні транзакції, інтеграцію з сайтом і підтримку різних методів оплати, що робить платіжну систему зручною та надійною.

XML-Sitemaps – інструмент для автоматичної генерації карти сайту. Карта сайту допомагає пошуковим системам ефективно індексувати сторінки проекту, що підвищує SEO-показники та видимість сайту у пошуку.

Кожен із цих інструментів було обрано на основі його функціональності, стабільності та сумісності з іншими технологіями проекту, що дозволило створити сучасний, надійний та масштабований веб-додаток.

3.2 Розробка автоматизованої системи з підбору комп’ютерних комплектуючих

Опишемо процес розробки основних модулів системи.

Спершу було проведено проєктування архітектури та моделювання даних. Для цього використовували Django ORM та Lucidchart для формалізації моделей Order, OrderLineItem, Product та UserProfile. Було визначено поля для зберігання інформації про замовлення, оплату, контактні дані користувачів та склад корзини, а також встановлено взаємозв’язки між таблицями для нормалізації даних. Після цього розроблялася основна логіка обробки замовлень та інтеграції з Stripe. Було створено клас StripeWH_Handler, який відповідає за обробку вебхуків Stripe. В його складі реалізовані три основні методи: handle_event для обробки загальних або непередбачуваних подій, що забезпечує стабільну роботу при отриманні невідомих вебхуків, handle_payment_intent_succeeded для обробки успішних платежів, який отримує дані про платіж і корзину користувача, очищає інформацію про доставку, оновлює профіль користувача за потреби, перевіряє наявність замовлення в базі даних із п’ятьма повторними спробами у разі затримки та створює нове замовлення разом із лініями замовлення на основі JSON-корзини, і handle_payment_intent_payment_failed для обробки невдалих платежів, що дозволяє фіксувати помилки та повторно обробляти вебхуки. Для інформування користувачів про успішне оформлення замовлення реалізовано метод _send_confirmation_email, який формує тему та текст листа на основі шаблонів Django і відправляє електронний лист через send_mail. Цей метод дозволяє динамічно включати інформацію про конкретне замовлення та гарантує отримання підтвердження навіть у разі повторної обробки вебхука. Обробка даних користувача та профілю забезпечує оновлення контактних даних та адреси доставки у разі, якщо користувач автентифікований

і відмітив опцію збереження інформації, що підвищує зручність повторних покупок (рис. 3.1).

```
def handle_event(self, event):
    """
    Handle a generic/unknown/unexpected webhook event
    """
    return HttpResponse(
        content=f'Unhandled webhook received: {event["type"]}',
        status=200)

def handle_payment_intent_succeeded(self, event):
    """
    Handle the payment_intent.succeeded webhook from Stripe
    """
    # Get payment intent
    intent = event.data.object
    # Get info from intent metadata
    pid = intent.id
    cart = intent.metadata.cart
    save_info = intent.metadata.save_info

    # Get the Charge object
    stripe_charge = stripe.Charge.retrieve(
        intent.latest_charge
    )

    billing_details = stripe_charge.billing_details
    shipping_details = intent.shipping
    grand_total = round(stripe_charge.amount / 100, 2)

    # Clean the data in the shipping details
    for field, value in shipping_details.address.items():
        if value == "":
            shipping_details.address[field] = None

    # Update profile information if save_info was checked
    # Profile set to None so we can still allow anonymous users to checkout
    profile = None
    username = intent.metadata.username
```

Рисунок 3.1 – Фрагмент розробки модуля webhook_handler.py

Для уникнення дублювання або пропуску замовлень реалізовано цикл із п’ятьма спробами перевірки наявності замовлення перед його створенням, з паузою в одну секунду між спробами, що підвищує надійність системи. Усі критичні операції щодо створення замовлень та ліній замовлення обгорнуті у блок try-except, і у разі виникнення виключень замовлення видаляється, а Stripe отримує статус 500 для автоматичного повторного виклику вебхука. Весь функціонал інтегрований із Django та динамічними шаблонами для формування HTML-контенту підтверджень замовлень та листів, що підключає дані моделей Order і UserProfile.

Розробка модуля оформлення замовлення та обробки платежів у цьому веб-проєкті на Django була побудована таким чином, щоб забезпечити повний

цикл обробки корзини користувача, створення замовлення, інтеграцію з платіжною системою Stripe та оновлення профілю користувача при необхідності. На початку модуль імпортує необхідні компоненти Django для роботи з HTTP-запитами, редиректами, повідомленнями, шаблонами та отриманням об’єктів з бази даних. Також підключені моделі замовлення та продуктів, форми користувачів і контекст корзини, що дозволяє взаємодіяти з даними профілю та вмістом кошика. Основний функціонал реалізовано у трьох функціях. Перша, cache_checkout_data, обробляє POST-запити для тимчасового збереження даних перед створенням платежу. Вона отримує ідентифікатор платежу Stripe з параметра client_secret, встановлює ключ API Stripe і модифікує метадані платежу, додаючи JSON-корзину, позначку про збереження інформації та ім’я користувача. У разі помилки надсилається повідомлення користувачу про неможливість обробки платежу, а сервер повертає HTTP-код 400. Друга функція, checkout, відповідає за відображення сторінки оформлення замовлення та обробку форми замовлення. Якщо корзина порожня, користувач перенаправляється назад до сторінки продуктів із відповідним повідомленням. У випадку POST-запиту форму OrderForm заповнюють даними з форми, перевіряють на валідність, присвоюють замовленню ідентифікатор Stripe, зберігають оригінальну корзину в JSON-форматі та зберігають об’єкт замовлення у базі даних. Для кожного продукту з корзини створюється відповідний OrderLineItem, що пов’язує замовлення з продуктом та кількістю. У разі відсутності продукту в базі даних замовлення видаляється, а користувач отримує повідомлення про необхідність звернення до підтримки. Далі зберігається інформація про опцію збереження даних користувача у сесії і відбувається перенаправлення на сторінку успішного оформлення замовлення. У разі GET-запиту модуль формує сторінку оформлення замовлення, попередньо перевіряючи наявність товарів у корзині. Створюється PaymentIntent у Stripe із сумою замовлення та валютою, після чого, якщо

користувач автентифікований, попередньо заповнюються поля форми даними з профілю користувача для зручності повторних покупок. У випадку відсутності публічного ключа Stripe виводиться попереджувальне повідомлення. Третя функція, checkout_success, обробляє сценарій успішного завершення платежу. Вона отримує номер замовлення, перевіряє опцію збереження даних користувача, приєднує замовлення до профілю користувача, а у разі активації збереження інформації оновлює дані профілю за допомогою форми UserProfileForm. Користувач отримує повідомлення про успішне оформлення замовлення з номером замовлення та інформацією про відправку підтвердження на електронну пошту. Після цього корзина видаляється зі сесії користувача для запобігання повторного оформлення тих самих товарів. Відображення сторінок реалізовано через рендеринг шаблонів checkout/checkout.html та checkout/checkout_success.html з передачею відповідних контекстних змінних, включаючи форму замовлення, публічний ключ Stripe, client_secret та об’єкт замовлення. Весь модуль побудований таким чином, щоб забезпечити надійну, безпечну і зручну для користувача роботу системи електронної комерції, дозволяючи обробляти замовлення, інтегруватися з платіжною системою, оновлювати профілі користувачів та інформувати їх про успішне завершення операцій (рис.3.2).

```
def checkout(request):
    stripe_public_key = settings.STRIPE_PUBLIC_KEY
    stripe_secret_key = settings.STRIPE_SECRET_KEY

    cart = request.session.get('cart', {})
    # Redirect if cart is empty
    if not cart:
        messages.error(request, "There's nothing in your cart at the moment")
        return redirect(reverse('products'))

    if request.method == 'POST':
        # Process checkout form submission
        cart = request.session.get('cart', {})
        form_data = {
            'full_name': request.POST['full_name'],
            'email': request.POST['email'],
            'phone_number': request.POST['phone_number'],
            'country': request.POST['country'],
            'postcode': request.POST['postcode'],
            'town_or_city': request.POST['town_or_city'],
            'street_address1': request.POST['street_address1'],
            'street_address2': request.POST['street_address2'],
            'county': request.POST['county'],
        }

        order_form = OrderForm(form_data)

        if order_form.is_valid():
            order = order_form.save(commit=False)
            pid = request.POST.get('client_secret').split('_secret')[0]
            order.stripe_pid = pid
            order.original_cart = json.dumps(cart)
            order.save()

            for item_id, item_data in cart.items():
                try:
                    product = Product.objects.get(id=item_id)
                    order_line_item = OrderLineItem(
```

Рисунок 3.2 – Фрагмент розробки модуля views.py

3.3 Висновки до розділу 3

У результаті проведеної розробки автоматизованої системи з підбору комп’ютерних комплектуючих було створено комплексну веб-платформу, яка інтегрує функціонал управління корзиною користувача, обробки замовлень, системи оплати та управління профілями користувачів. Вибір сучасного стека технологій, включно з Django, Python, Stripe, PostgreSQL, Bootstrap та іншими інструментами, забезпечив високу надійність, безпеку та масштабованість системи, дозволив спростити процес розробки та подальшої підтримки проекту. Процес розробки включав проєктування архітектури даних, формування моделей та зв’язків між ними, реалізацію логіки обробки замовлень через модулі webhook_handler.py і views.py, а також інтеграцію динамічного рендерингу HTML-контенту за допомогою шаблонів Django. Впроваджені механізми перевірки наявності замовлення, обробки помилок та повторних спроб створення замовлень підвищують надійність роботи системи та запобігають дублюванню даних. Інтеграція з платіжною системою Stripe та обробка вебхуків дозволяє забезпечити безпечну та оперативну оплату товарів, а функціонал збереження інформації користувача підвищує зручність повторних покупок. Використання динамічних форм, адаптивного дизайну та візуальних елементів підвищує користувацький досвід та зручність роботи з сайтом. У підсумку, розроблена система демонструє повний цикл автоматизації процесу підбору і замовлення комп’ютерних комплектуючих, поєднуючи сучасні технології для забезпечення безперебійної, безпечної та ефективної роботи веб-платформи, що робить її придатною для реального комерційного використання та подальшого масштабування.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Тестування автоматизованої системи з підбору комп'ютерних комплектуючих

Поетапно розглянемо основні опції та вікна розробленої системи. Вікно головної сторінки має наступний вигляд (рис.4.1).

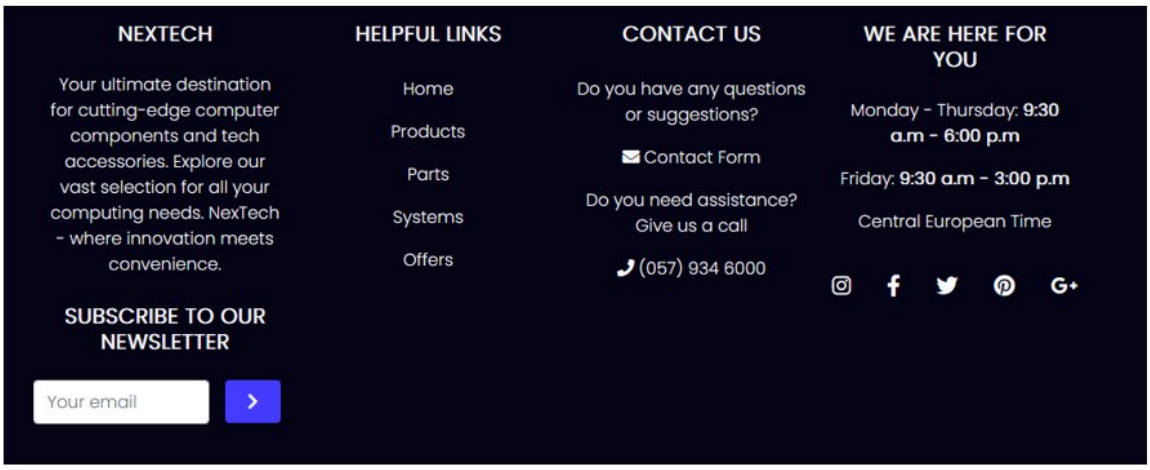


Рисунок 4.1 – Хедер головної сторінки

Сторінка реєстрації в системі виглядає наступним чином (рис.4.2).

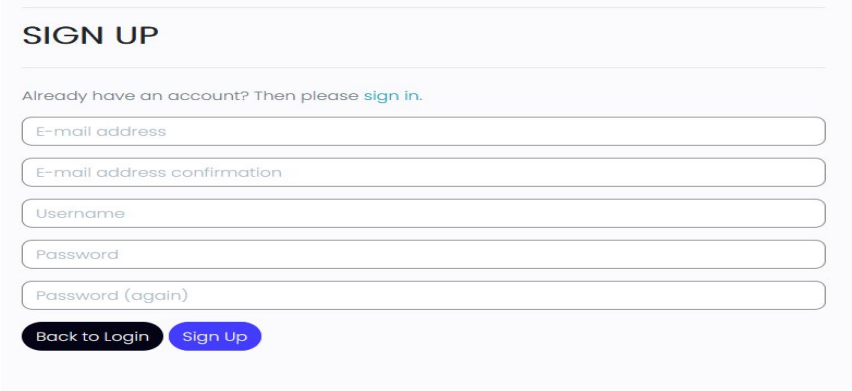


Рисунок 4.2 – Сторінка реєстрації в системі

Надалі перейдемо на сторінку власне комплектуючих, доступна фільтрація за алфавітом по категоріям, рейтингом та цінами (рис.4.3).

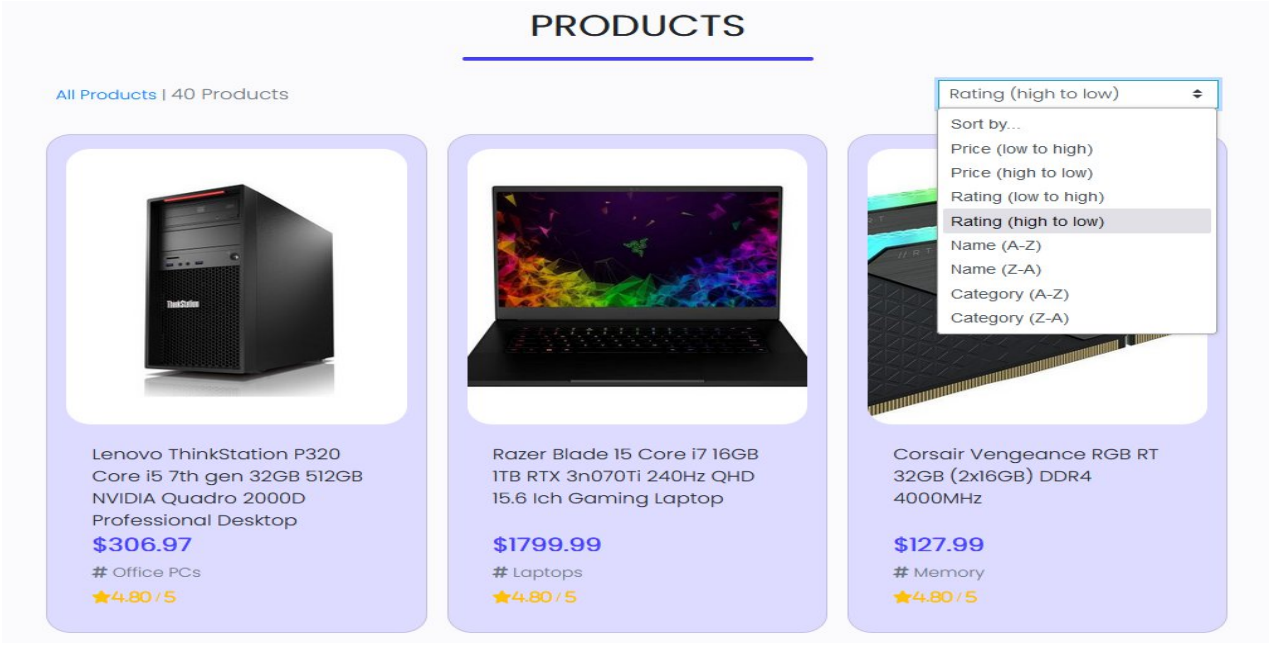


Рисунок 4.3 – Основна сторінка комплектуючих з фільтрами

Обравши певний компонент, можна прочитати про нього детальніше та додати в кошик (рис.4.4).

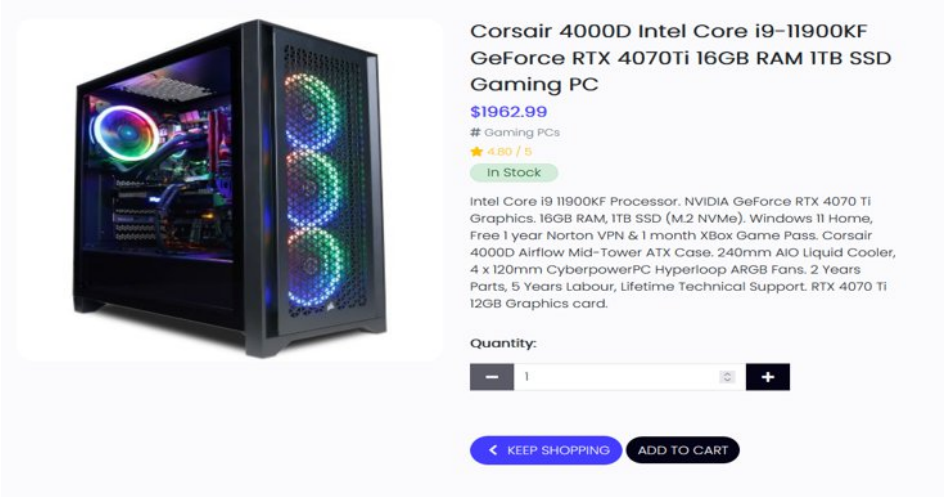


Рисунок 4.4 – Сторінка окремого комплектуючого компонента

Користувач має змогу сплатити за замовлення одразу та побачити відомості про його доставку (рис.4.5).

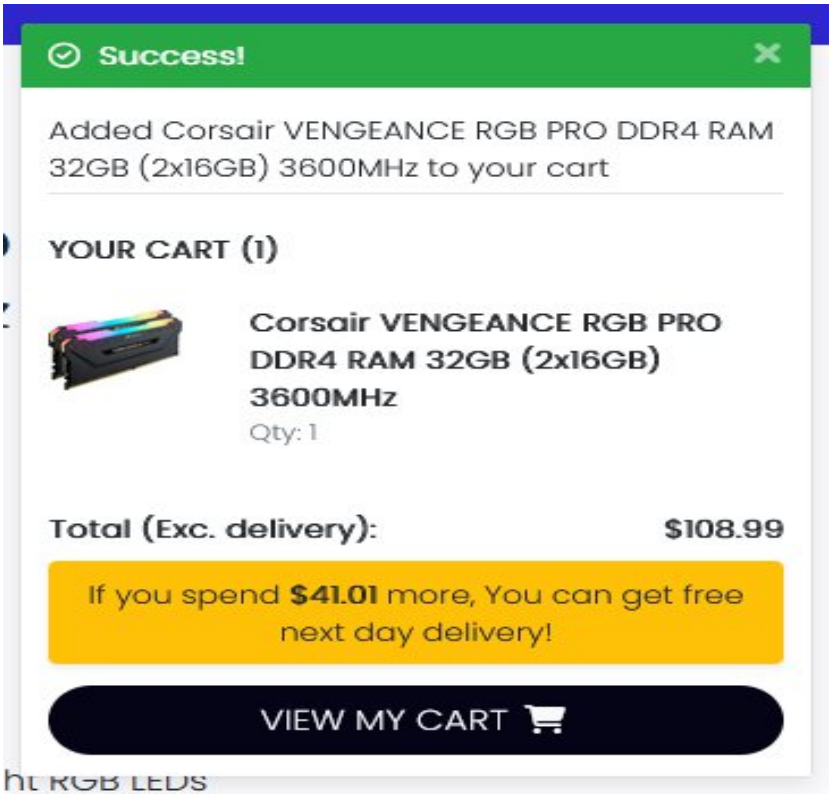


Рисунок 4.5 – Оплата замовлення

Також є можливість додавання товарів до вішліста (рис.4.6).

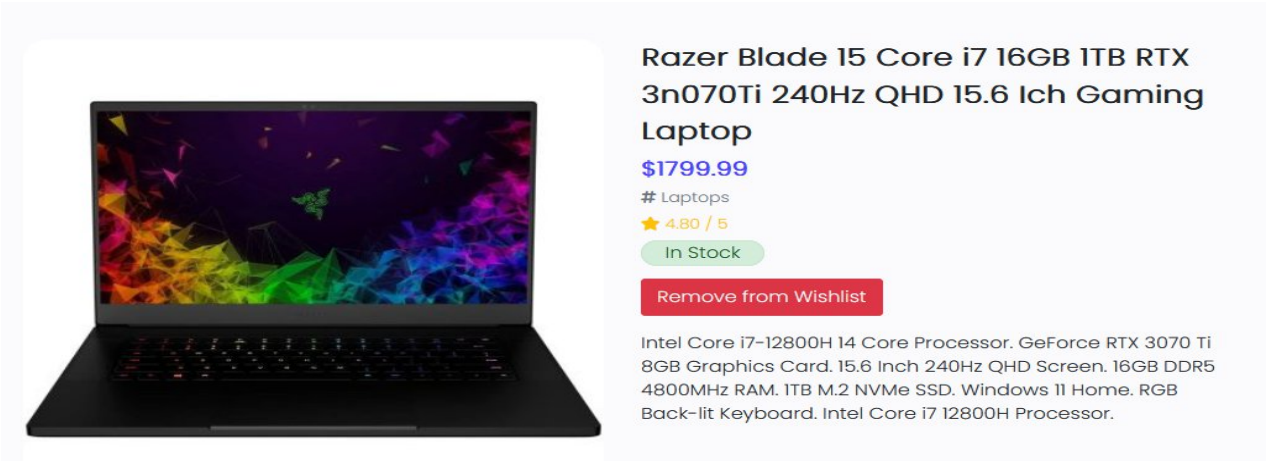


Рисунок 4.6 – Додавання товарів до вішліста

Редагування опису товарів на панелі адміна має наступний вигляд (рис.4.7).

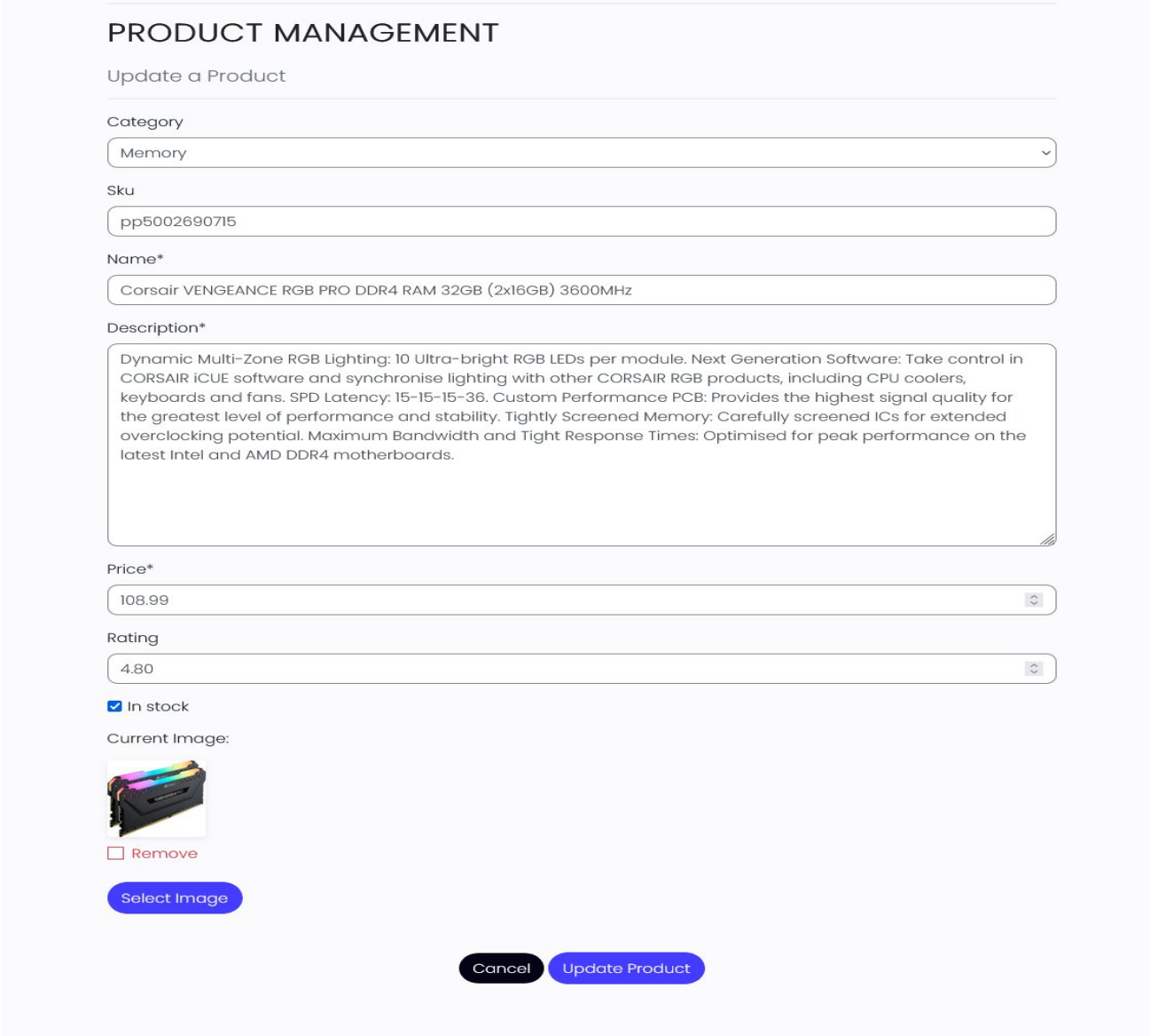


Рисунок 4.7 – Редагування опису товарів адміном

Головна домашня сторінка виглядає яскраво та привабливо з маркетингової точки зору (рис.4.8).

Рисунок 4.8 – Головна домашня сторінка додатку

Оброблення замовлення перед оплатою є деталізованим та дозволяє обрати потрібні товари з дода них у корзину (рис.4.9).

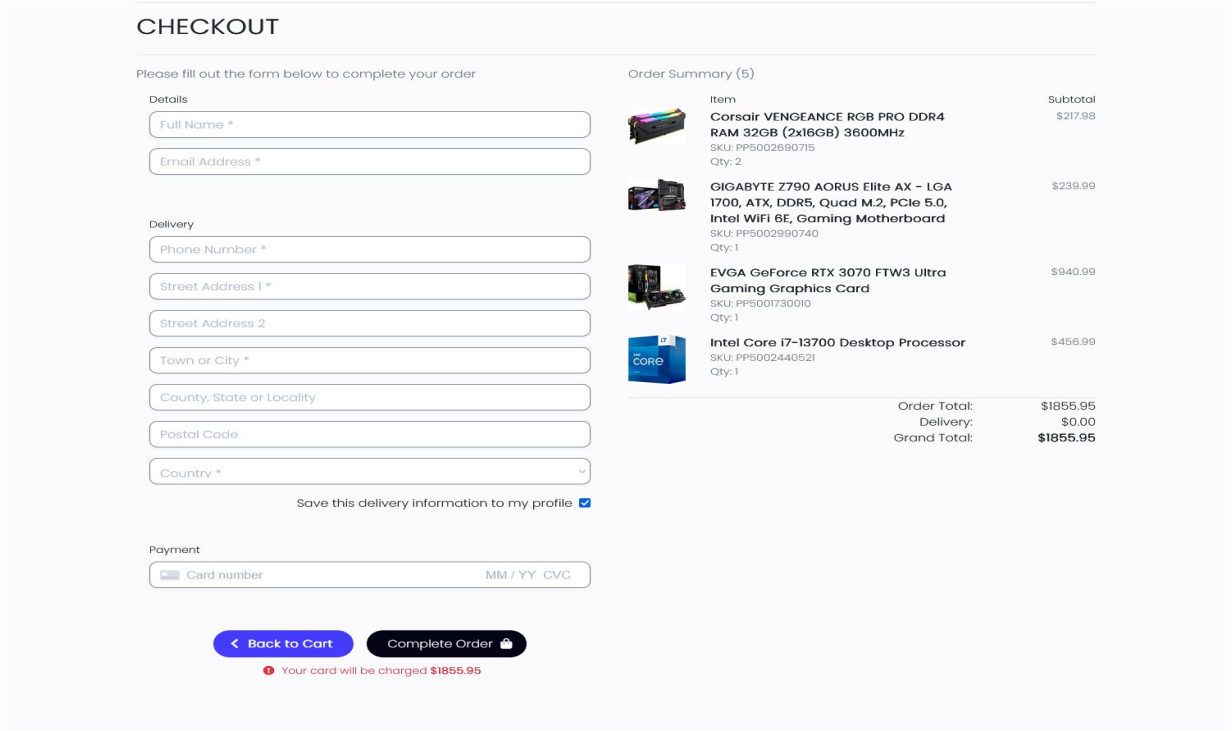


Рисунок 4.9 – Оброблення замолвення перед оплатою

Квитанція замовлення містить всю неообхідну інформацію про отримувача та про замовлення (рис. 4.10).

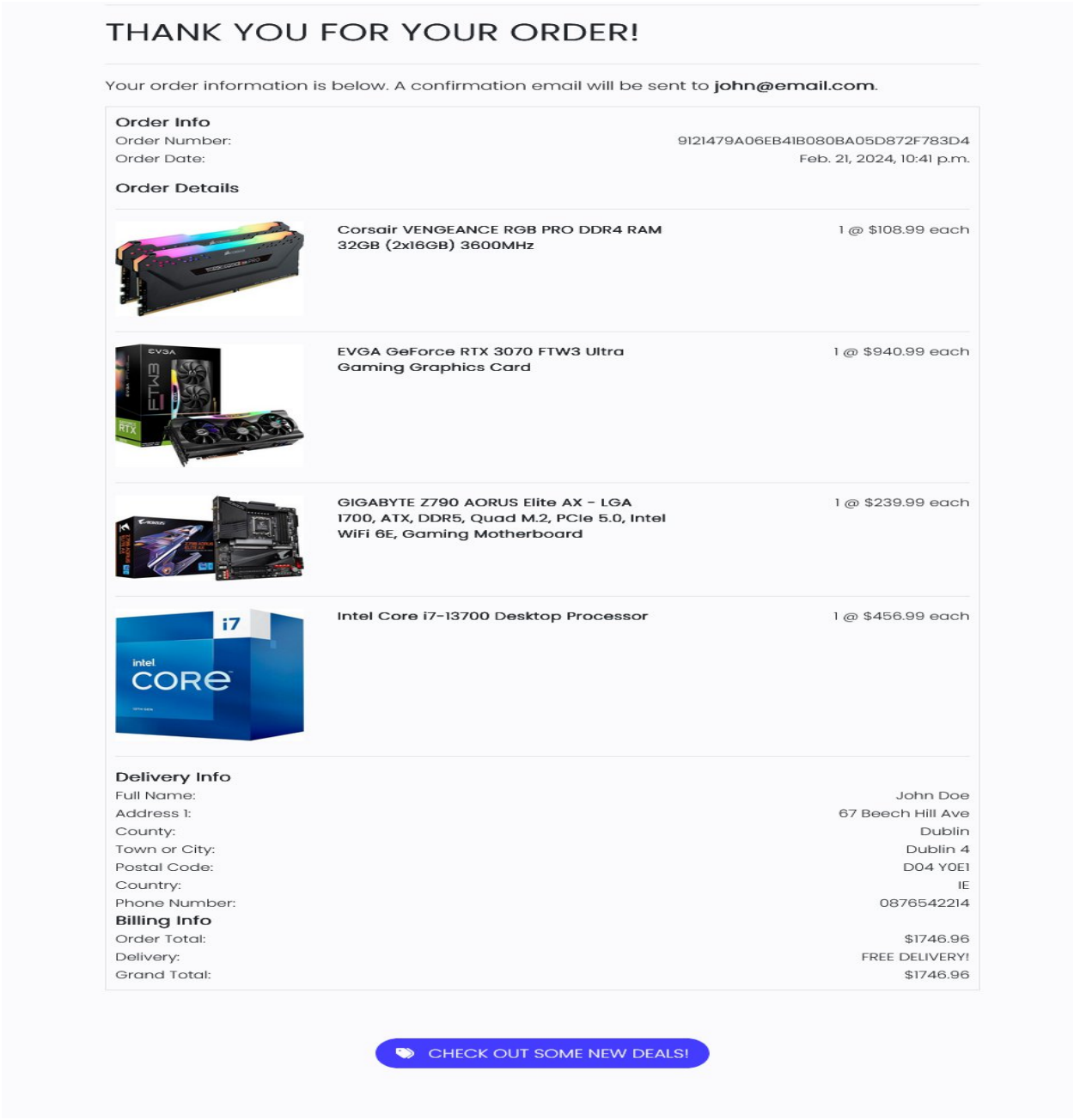


Рисунок 4.10 – Квитанція замовлення

Профіль адміністратора містить деталізовану інформацію про всі покупки та замовників (рис. 4.11).

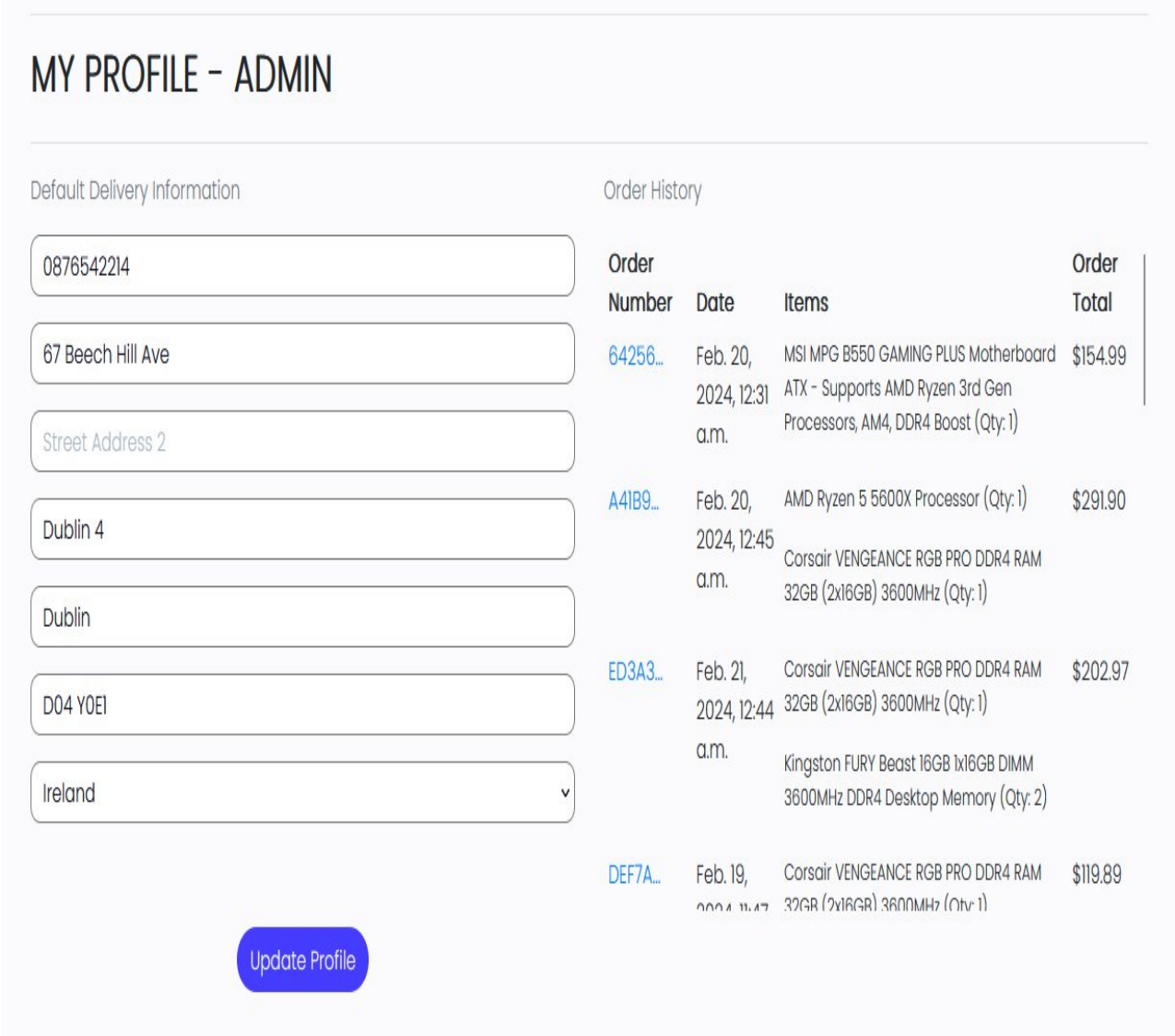


Рисунок 4.11 – Деталізована інформація в профілі адміністратора про всі замовлення

На рисунку 4.12 показано, як можна додати та видалити замовлення в список бажаних.

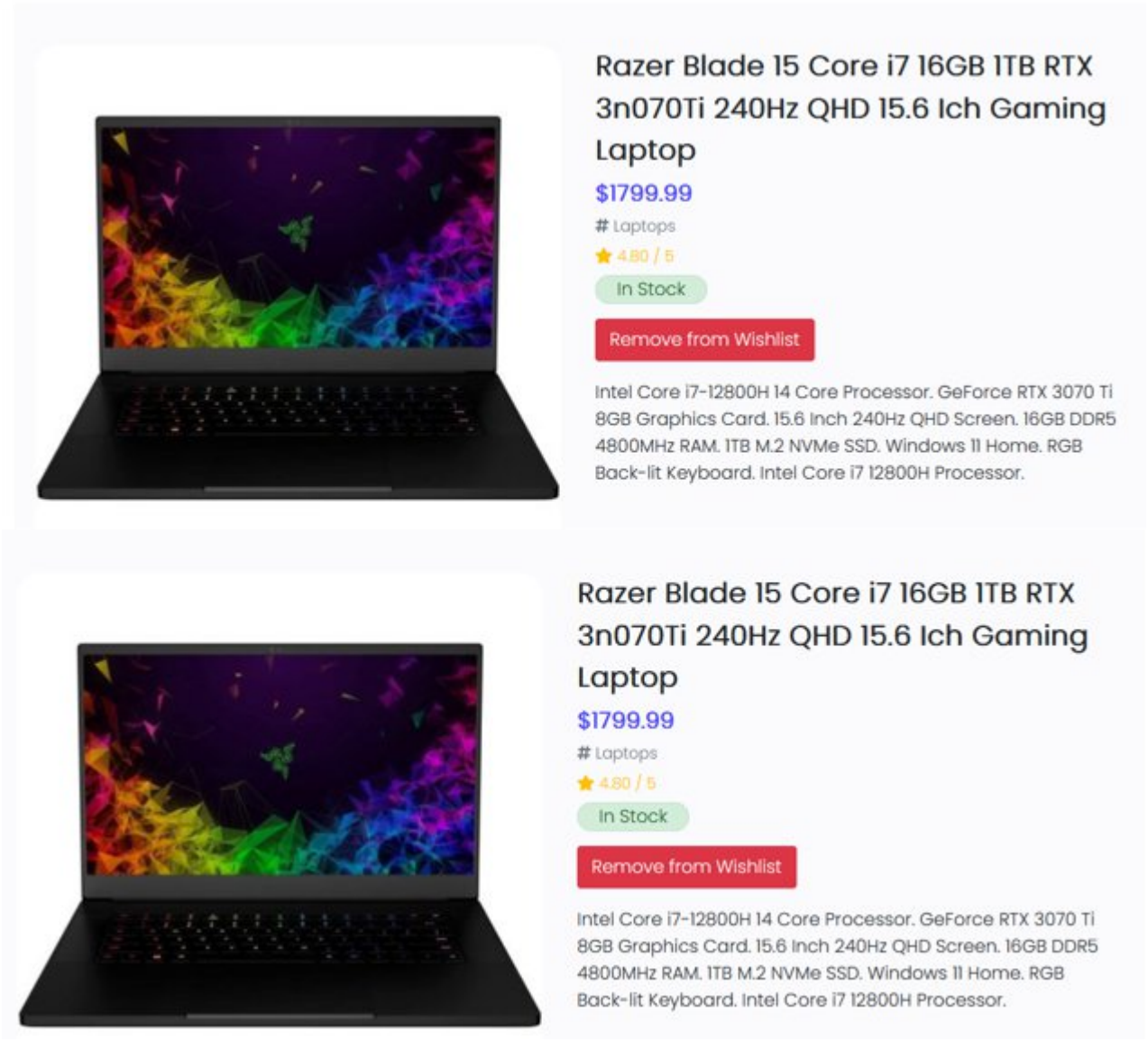


Рисунок 4.12 – Додавання товарів до вішліста та видалення з нього

На рисунку 4.13 показано форму, яку заповнює адміністратор при додаванні нового товару.

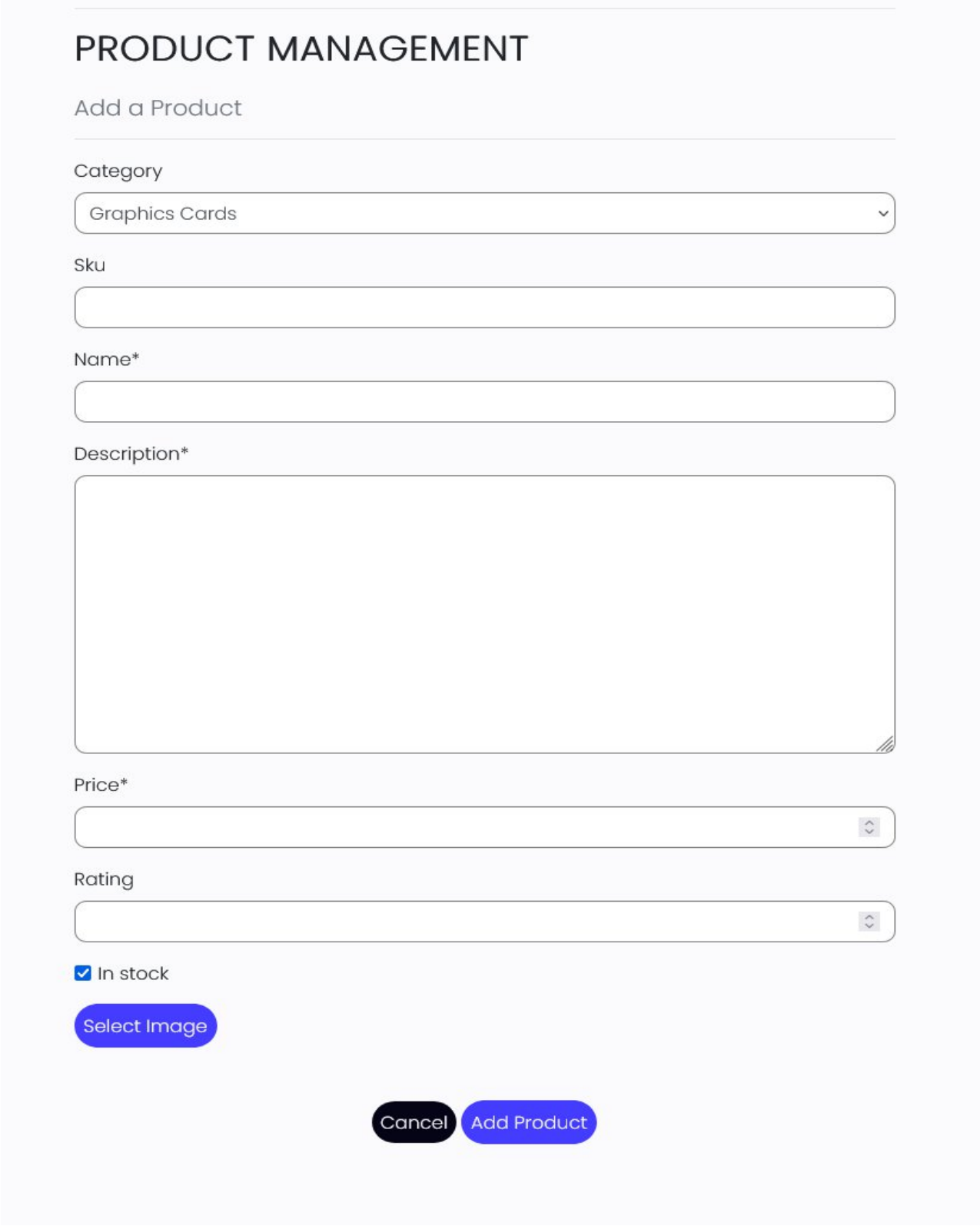


Рисунок 4.13 – Форма додавання нового товару в профілі адміністратора

В таблиці 4.1 наведено результати повного тестування автоматизованої системи з підбору комплектуючих.

Таблиця 4.1 – Результати повного тестування автоматизованої системи з підбору комплектуючих

Історія	**Тест**	**Очікуваний результат**	**Результат**
Створення облікового запису	Реєстрація через меню акаунта	Користувач може ввести інформацію у форму реєстрації з відповідною валідацією	пройдено
Підтвердження email	Реєстрація через меню акаунта	Користувач отримує лист для підтвердження email та успішно завершує реєстрацію	пройдено
Вхід/Вихід	Вхід та вихід з акаунта	Процес входу та виходу з акаунта виконується успішно	пройдено
Скидання паролю	Клік на посилання для скидання паролю на сторінці входу	Користувач отримує лист для скидання паролю	пройдено
Профіль користувача	Вхід на сайт та доступ до сторінки профілю через меню акаунта	Користувач може переглядати профіль, історію замовлень та зберігати/оновлювати	пройдено

		дані доставки	
Перегляд та навігація	Список продуктів	Перехід на сторінку продуктів відображає всі товари у вигляді карток з інформацією	пройдено
Перегляд без акаунта	Перехід на сторінку продуктів без входу	Всі товари та їх інформація відображаються у вигляді карток	пройдено
Акції та спеціальні пропозиції	Відкрити меню пропозицій	Відображаються товари категорій Новинки, Акції та Розпродаж	пройдено
Загальна сума	Додати товари у кошик	Загальна вартість замовлення динамічно оновлюється на всіх сторінках	пройдено
Деталі продукту	Клік на зображення товару	Відкривається сторінка з детальною інформацією про товар: зображення, назва, ціна, категорія, рейтинг, наявність, опис та кількість	пройдено
Сортування та пошук	Фільтри	Вибрані критерії сортування відображаються у	пройдено

		списку товарів	
Категорії продуктів	Вибір категорії з меню	Відображаються товари, що відповідають обраній категорії	пройдено
Пошук у категорії	Вибір категорії + фільтр сортування	Відображаються відсортовані товари обраної категорії	пройдено
Пошук продукту	Введення запиту у форму пошуку	Відображаються товари, чиї назви або опис відповідають запиту	пройдено
Перегляд результатів пошуку та фільтрування	Фільтрування та пошук одночасно	Товари відображаються відповідно до обраних фільтрів та пошуку	пройдено
Список бажаних товарів	Додавання/видалення товарів до wishlist	Кнопки wishlist доступні для зареєстрованих користувачів, товари можна додавати/видаляти	пройдено
Контакт та комунікація	Відкриття та заповнення контактної форми	Форма контактів працює та дозволяє відправляти дані з валідацією	пройдено
Підписка на	Введення email для	Користувач може	пройдено

розсилку	розсилки	підписатися на розсилку, отримує повідомлення про успіх	
Покупка та оформлення	Вибір кількості товару	Кнопки змінюють кількість товару, зміни відображаються у кошику та сумі замовлення з валідацією	пройдено
Додавання у кошик	Додавання товару через сторінку продукту	Товар додається у кошик, відображається повідомлення про успіх та оновлюється загальна сума	пройдено
Зміна кількості у кошику	Зміна кількості у кошику	Кількість змінюється, загальна сума оновлюється, є валідація	пройдено
Безпечна та зручна оплата	Проходження оформлення з Stripe тестовою карткою	Оформлення завершується успішно, користувач перенаправляється на сторінку підтвердження	пройдено
Підтвердження замовлення	Завершення оформлення	Користувач бачить деталі замовлення,	пройдено

		товари, дані доставки та оплати	
Підтвердження email замовлення	Завершення оформлення	Користувач отримує лист із підтвердженням замовлення та деталями доставки та оплати	пройдено
Адмін та управління магазином	Додавання товарів	Адмін може додати новий товар через форму з валідацією	пройдено
Редагування товарів	Редагування товару через кнопку редагування	Адмін може редагувати товар, зміни зберігаються та відображаються на сайті	пройдено
Видалення товарів	Видалення товару через кнопку видалення	Адмін може видаляти товар, підтвердження через модальне вікно	пройдено

4.2 Аналіз результатів тестування автоматизовної системи з підбору комп’ютерних комплектуючих

На основі результатів, наведених у таблиці 4.1, було проведено комплексне повне тестування автоматизованої системи з підбору комплектуючих, яке охопило всі ключові функціональні модулі системи, користувацькі сценарії та адміністративні процеси. Тестування виконувалося з позиції кінцевого користувача, неавторизованого відвідувача та адміністратора

системи, що дозволило перевірити коректність роботи системи в реальних умовах експлуатації.

На етапі тестування підсистеми автентифікації та керування обліковими записами було перевірено процес створення нового користувацького акаунта через меню облікового запису. Встановлено, що форма реєстрації коректно приймає дані користувача, виконує валідацію обов'язкових полів, перевіряє формат електронної пошти та пароля, а також блокує збереження некоректно введеної інформації. Після реєстрації успішно відпрацьовує механізм підтвердження електронної пошти: користувач отримує автоматичний лист із посиланням для активації акаунта, після переходу за яким реєстрація завершується без помилок. Додатково було протестовано процеси входу та виходу з облікового запису, які виконуються стабільно, без втрати сесії або некоректної авторизації. Функція скидання пароля також показала коректну роботу: після натискання відповідного посилання користувач отримує лист для відновлення доступу, а новий пароль успішно зберігається в системі.

Додатково було перевірено функціональність входу та виходу з облікового запису, включаючи обробку коректних і некоректних облікових даних, збереження сесії користувача та коректне її завершення після виходу. Функція скидання пароля також була протестована окремо: при натисканні на відповідне посилання на сторінці входу система надсилає користувачеві електронний лист для відновлення доступу, після чого новий пароль успішно зберігається та дозволяє виконати повторний вхід до системи.

В межах тестування персонального кабінету користувача було перевірено доступ до сторінки профілю після авторизації. Встановлено, що система коректно відображає персональні дані користувача, історію попередніх замовлень, а також інформацію щодо доставки. Функції редагування та збереження цих даних працюють стабільно, зміни коректно фіксуються у базі даних та відображаються при повторному вході до системи.

Значну частину тестування було присвячено перевірці каталогу товарів та навігації сайтом. Було встановлено, що при переході на сторінку продуктів система відображає повний перелік доступних товарів у вигляді візуально структурованих карток, які містять усю необхідну базову інформацію. Окремо підтверджено, що перегляд товарів доступний і для неавторизованих користувачів, що забезпечує відкритість системи та зручність первинного ознайомлення з асортиментом. Розділи акцій, новинок та розпродажів також функціонують коректно, відображаючи відповідні категорії товарів без затримок і помилок.

Під час тестування сторінки детального перегляду продукту було підтверджено, що при переході з каталогу відкривається сторінка з повною та актуальною інформацією про товар, включаючи зображення, назву, ціну, категорію, рейтинг, стан наявності, опис та можливість вибору кількості одиниць. Усі елементи інтерфейсу працюють узгоджено, а відображені дані відповідають інформації, що зберігається в системі.

Окремий блок тестування був присвячений функціям пошуку, сортування та фільтрації товарів. У ході перевірки встановлено, що користувач може ефективно здійснювати пошук за ключовими словами, а система коректно відображає товари, назви або описи яких відповідають введеному запиту. Фільтрація за категоріями та параметрами сортування застосовується без помилок, результати оновлюються динамічно, а поєднання декількох фільтрів одночасно не призводить до логічних конфліктів або некоректного відображення даних.

Функціонал списку бажаних товарів було протестовано для зареєстрованих користувачів. Встановлено, що система дозволяє додавати та видаляти товари з wishlist, зберігає обрані позиції та забезпечує їх доступність при повторних сесіях користувача.

У процесі тестування комунікаційних можливостей системи перевірено роботу контактної форми та механізму підписки на розсилку. Контактна форма коректно обробляє введені дані, виконує їх валідацію та забезпечує успішну передачу повідомлень. Аналогічно, функція підписки на розсилку дозволяє користувачеві ввести email-адресу, отримати підтвердження про успішну підписку та забезпечує збереження даних у системі.

Найбільш критичним етапом тестування став процес оформлення замовлення та оплати. Було підтверджено коректну роботу додавання товарів у кошик, зміни їх кількості та динамічного оновлення загальної вартості замовлення на всіх етапах взаємодії. Процес оформлення покупки з використанням платіжної системи Stripe у тестовому режимі завершується успішно, після чого користувач перенаправляється на сторінку підтвердження замовлення. На цій сторінці відображаються всі ключові деталі, включаючи перелік товарів, дані доставки та інформацію про оплату. Додатково підтверджено коректне надсилання електронного листа з підтвердженням замовлення.

Подальше тестування було зосереджене на перевірці логіки класифікації товарів, інтелектуального пошуку, персоналізованих користувацьких функцій, а також засобів комунікації між користувачем і системою. Особлива увага приділялася коректності обробки запитів, узгодженості результатів відображення та стабільності роботи інтерфейсу при комбінуванні декількох сценаріїв взаємодії одночасно.

В межах перевірки функціоналу категорій продуктів було протестовано механізм вибору категорії з головного меню сайту. У результаті тестування встановлено, що при переході до будь-якої обраної категорії система коректно формує вибірку товарів відповідно до заданої класифікації. Відображаються виключно ті позиції, які належать до обраної категорії, без домішок сторонніх товарів або логічних помилок. Перехід між категоріями здійснюється без

затримок, інтерфейс оновлюється коректно, а кількість товарів у кожній категорії відповідає фактичним даним, що зберігаються в системі.

Наступним етапом було детально перевірено пошук товарів в межах обраної категорії з одночасним застосуванням фільтрів сортування. У процесі тестування підтверджено, що після вибору категорії користувач може додатково застосовувати параметри сортування, такі як ціна, популярність або інші доступні критерії, і система коректно відображає відсортований перелік товарів лише в межах цієї категорії. Зміна параметрів сортування не призводить до втрати обраної категорії, а результати оновлюються динамічно та логічно, що свідчить про правильну реалізацію взаємодії між модулями фільтрації та категоризації.

Окремо було протестовано загальний механізм пошуку продуктів через форму пошуку. У ході перевірки встановлено, що система коректно обробляє текстові запити користувача та формує результати на основі відповідності назв або описів товарів введеним ключовим словам. Пошук працює стабільно як для коротких, так і для розширених запитів, а результати відображаються коректно без дублювання або пропуску релевантних позицій. За відсутності відповідних товарів система коректно інформує користувача, не порушуючи загальної логіки навігації.

Особливу увагу було приділено перевірці одночасного використання пошуку та фільтрації. В межах цього сценарію користувач може застосовувати фільтри та виконувати пошукові запити паралельно, при цьому система коректно поєднує обидва механізми та відображає лише ті товари, які одночасно відповідають і пошуковому запиту, і встановленим фільтрам. У ході тестування підтверджено, що комбінування декількох умов не призводить до конфліктів, некоректних вибірок або помилок відображення, а результати залишаються логічно узгодженими та зрозумілими для користувача.

Подальше тестування охоплювало функціонал списку бажаних товарів. Було підтверджено, що кнопки додавання та видалення товарів із wishlist доступні виключно для авторизованих користувачів, що відповідає вимогам безпеки та персоналізації. У процесі взаємодії з цією функцією товари коректно додаються до списку бажаних, зберігаються у профілі користувача та можуть бути видалені без втрати інших даних. Стан списку бажаних коректно зберігається між сесіями, що підтверджує стабільну роботу механізму персональних збережень.

В межах тестування комунікаційного блоку системи було перевірено роботу контактної форми. Під час відкриття форми користувачеві надається можливість заповнити всі необхідні поля, при цьому система здійснює перевірку коректності введених даних та не дозволяє відправлення повідомлення у разі порушення правил валідації. Після введення коректної інформації форма успішно надсилає повідомлення, що підтверджує працездатність каналу зворотного зв’язку між користувачем і системою.

Подальший етап тестування був присвячений перевірці найбільш критичних для практичного використання системи функцій, а саме процесів покупки, оформлення замовлення, оплати, підтвердження результатів транзакції та адміністративного управління товарним наповненням. Саме ці модулі визначають комерційну цінність автоматизованої системи з підбору комплектуючих та безпосередньо впливають на довіру користувачів і стабільність її експлуатації.

В межах тестування процесу покупки було детально перевірено механізм вибору кількості товару. Встановлено, що кнопки збільшення та зменшення кількості одиниць товару працюють коректно як на сторінці детального перегляду продукту, так і в кошику. Будь-які зміни кількості миттєво відображаються у вмісті кошика та загальній сумі замовлення. Система коректно обробляє граничні значення, не допускає введення від’ємних чисел

або невалідних значень, що свідчить про наявність вбудованих механізмів валідації.

Окремо було протестовано додавання товарів у кошик безпосередньо зі сторінки продукту. Після виконання цієї дії система коректно додає обраний товар до кошика, відображає відповідне повідомлення про успішну операцію та оновлює загальну суму замовлення. Вміст кошика зберігається при переході між сторінками, що підтверджує стабільну роботу механізму збереження стану користувацької сесії.

Зміна кількості товарів безпосередньо у кошику також була перевірена окремим сценарієм. У результаті тестування встановлено, що при зміні кількості кожної позиції система автоматично перераховує вартість як окремого товару, так і всього замовлення загалом. Усі зміни супроводжуються коректною валідацією та не призводять до логічних помилок або розбіжностей у підсумкових сумах.

Ключовим елементом тестування став процес оформлення замовлення та здійснення оплати. Було перевірено повний сценарій оформлення покупки із використанням платіжної системи Stripe в тестовому режимі. У ході тестування підтверджено, що система коректно передає дані замовлення до платіжного сервісу, обробляє результат транзакції та у разі успішної оплати автоматично перенаправляє користувача на сторінку підтвердження замовлення. Жодних збоїв або втрати даних під час цього процесу зафіксовано не було.

Після завершення оформлення замовлення користувачеві відображається сторінка підтвердження, на якій у структурованому вигляді подано всю ключову інформацію щодо покупки, зокрема перелік замовлених товарів, їх кількість, загальну вартість, дані доставки та інформацію про оплату. Дані на сторінці підтвердження повністю відповідають інформації, введеній користувачем та збереженій у системі. Додатково підтверджено, що після оформлення замовлення система автоматично надсилає електронний лист із підтвердженням

покупки, який містить усі необхідні деталі та слугує офіційним підтвердженням транзакції для користувача.

Функція редагування товарів також була протестована детально. У ході перевірки підтверджено, що адміністратор може змінювати характеристики товару, зберігати оновлені дані та бачити їх коректне відображення на сайті без необхідності додаткових дій. Видалення товарів із каталогу здійснюється через відповідну кнопку з підтвердженням дії у модальному вікні, що мінімізує ризик випадкового видалення даних. Після підтвердження товар коректно видаляється з системи та більше не відображається у каталозі.

Завершальний блок тестування було зосереджено виключно на перевірці адміністративних можливостей системи, оскільки саме цей функціонал забезпечує актуальність товарного каталогу, керованість контенту та безперервність коректної роботи всієї автоматизованої системи з підбору комплектуючих. На даному етапі тестування система розглядалася з точки зору адміністратора, який відповідає за наповнення, підтримку та оновлення інформації про товари.

У процесі тестування функції додавання нових товарів було детально перевірено роботу адміністративної форми створення продукту. Адміністратор має змогу вводити повний набір характеристик товару, включаючи назву, опис, ціну, категорію, наявність, зображення та інші параметри, передбачені логікою системи. Під час введення даних система здійснює контроль обов’язкових полів, перевіряє коректність числових значень і не дозволяє зберегти товар у разі відсутності або помилковості критично важливої інформації. Після успішного заповнення форми новий товар коректно зберігається в базі даних і без затримок з’являється в користувацькому каталозі, що підтверджує правильну інтеграцію адміністративного модуля з публічною частиною системи.

Окрему увагу було приділено тестуванню редагування існуючих товарів. В межах цього сценарію адміністратор отримує доступ до вже створених позицій та може змінювати будь-які їхні параметри без необхідності повторного

створення товару. У ході тестування підтверджено, що система коректно завантажує актуальні дані товару в форму редагування, дозволяє вносити зміни та забезпечує їх коректне збереження. Після збереження оновлена інформація миттєво відображається у каталозі для кінцевих користувачів, без порушення структури сторінок або втрати пов’язаних даних, таких як категорія чи зображення. Це свідчить про стабільну роботу механізму оновлення інформації та відсутність конфліктів між старими й новими значеннями.

Функція видалення товарів була протестована з точки зору безпеки та запобігання випадковим діям. У результаті тестування встановлено, що при спробі видалення товару система ініціює додаткове підтвердження дії через модальне вікно, яке інформує адміністратора про наслідки операції. Лише після підтвердження товар остаточно видаляється з бази даних і більше не відображається ні в адміністративній частині, ні в публічному каталозі.

4.3 Висновки до розділу 4

Таким чином, тестування адміністративного функціоналу підтвердило, що система забезпечує повний, логічно узгоджений та безпечний набір інструментів для управління товарним каталогом. Додавання, редагування та видалення товарів реалізовані коректно, працюють стабільно та не порушують цілісності даних, що є критично важливим для подальшої експлуатації автоматизованої системи в умовах реального електронного комерційного середовища.

В результаті, внаслідок тестування доведено що система повністю працездатна та коректно розроблена.

5 ЕКОНОМІЧНИЙ РОЗДІЛ

5.1 Технологічний аудит результатів проведених наукових досліджень побудови автоматизованих систем з підбору комп’ютерних комплектуючих

Як було зазначено раніше, автоматизовані системи з підбору комп'ютерних комплектуючих — це програмні інструменти (онлайн-конфігуратори), які допомагають користувачам самостійно вибрати сумісні компоненти для збирання або модернізації персонального комп’ютера, автоматично перевіряючи їхню працездатність та узгодженість.

Для побудови таких систем, тобто створення ефективної автоматизованої системи підбору комплектуючих, під час проведення досліджень перед нами було поставлено низку ключових задач: а) дослідити існуючі системи збору і актуалізації даних, тобто проаналізувати інформацію про технічні, економічні та надійнісні характеристики тисяч компонентів ПК від різних виробників (процесори, материнські плати, відеокарти, блоки живлення, пам'ять тощо); б) провести моделювання сумісності, тобто розробити алгоритми та правила, які б автоматично перевіряли фізичну та електричну сумісність обраних компонентів; в) зробити класифікацію комплектуючих; г) забезпечити врахування вимог користувачів (наприклад, «комп'ютер для ігор», «для офісної роботи» тощо); розробити зрозумілий і зручний інтерфейс (онлайн-конструктор) для взаємодії з користувачем; розробити відповідне програмне забезпечення тощо.

В результаті виконаної МКР та проведених досліджень були отримані результати, які можуть дати користувачам готовий, збалансований та працездатний перелік комплектуючих для ПК, що відповідає їх потребам та бюджету

Для встановлення рівня комерційного потенціалу результатів проведених нами досліджень було проведено їх технологічний аудит. Для цього було запрошено 3-х експертів, кандидатів технічних наук, доцентів Гармаша В.В., Богач І.В. та Кулика Я.А.

Експерти здійснювали технологічний аудит результатів проведених досліджень згідно з критеріями, наведеними в таблиці 5.1 [25].

Таблиця 5.1 – Критерії за якими здійснювався технологічний аудит результатів проведених наукових досліджень

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Ринкові перспективи					
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Запрошені експерти оцінили результати проведених нами досліджень за 5-ти бальною шкалою («0», «1», «2», «3» та «4»), які зведено в таблицю 5.2.

Таблиця 5.2 – Результати технологічного аудиту отриманих наукових результатів

Критерії	Прізвище, ініціали експерта		
	Гармаш В.В.	Богач І.В.	Кулик Я.А.
	Бали, виставлені експертами:		
1	4	4	3
2	3	3	3
3	4	4	4
4	3	3	4
5	3	4	3
6	3	3	4
7	3	4	3
8	3	3	4
9	3	3	4
10	3	3	3
11	3	3	3
12	4	3	3
Сума балів	$СБ_1 = 39$	$СБ_2 = 40$	$СБ_3 = 41$

Далі розрахуємо середньоарифметичну суму балів, що їх виставили експерти:

Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{3} = \frac{39+40+41}{3} = \frac{120}{3} = 40.$

Керуючись рекомендаціями, наведеними а таблиці 5.3, можна зробити висновок, що, оскільки середньоарифметична сума балів, що їх виставили експерти, дорівнює 40-м балам, то результати проведених нами досліджень мають рівень комерційного потенціалу, який практично можна вважати «вище середнього».

Таблиця 5.3 – Рівні комерційного потенціалу отриманих наукових результатів

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

5.2 Розрахунок витрат на проведення наукових досліджень побудови автоматизованих систем з підбору комп'ютерних комплектуючих

При проведенні досліджень були зроблені такі основні витрати:

а). Основна заробітна плата $З_о$ дослідників, розробників, консультантів, фахівців тощо, величина якої визначається за формулою:

$$З_о = \frac{М}{Т_р} \times t \text{ грн}, \qquad (5.1)$$

де М – місячний посадовий оклад розробника (дослідника), грн;

Для 2025 року приймемо, що:

М = (8000…33500) грн/місяць;

$T_р$ – число робочих днів в місяці; приймемо $T_р$ = 25 днів;

t – число днів роботи розробників, дослідників, інших фахівців.

Зроблені розрахунки величини основної заробітної плати розробників, дослідників тощо зведемо до таблиці 5.4.

Таблиця 5.4 – Основна заробітна плата розробників (дослідників)

Найменування посади виконавця	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів (годин) роботи	Витрати на оплату праці, грн
1. Науковий керівник магістерської роботи, доцент	25400	1016,00	20 годин	(1016,00 / 6) × 20 = = 3386,66≈ 3387 (при 6-годинному робочому дні)
2. Здобувач-магістрант (виконавець)	8000	320,00	75 днів	24000
3. Інші фахівці	30000	1200	5 днів	6000 грн (при 8-годинному робочому дні)
4. Консультант з економічної частини	20500	820	1,5 години	(820,00 /6) × 1,5 = ≈ 205 грн (при 6-годинному робочому дні)
Загалом				$З_о$ = 33592 грн

Примітка. За заробітну плату магістранта приймемо мінімальну заробітну плату в країні

б). Додаткова заробітна плата $З_д$ розробників (дослідників), яка розраховується як (10…14)% від величини їх основної заробітної плати, тобто:

$$З_д = a \times З_о = (0,1...0,14) \times З_о . \quad (5.2)$$

Приймемо, що $\alpha = 0,14$. Тоді для нашого випадку отримаємо:

$$З_д = 0,14 \times 33592 = 4702,88 \approx 4703 \text{ грн.}$$

в). Нарахування на заробітну плату $НЗП_{зп}$ розробників (дослідників) та інших розраховуються за формулою:

$$НЗП_{зп} = (З_о + З_д) \times \frac{b}{100}, \quad (5.3)$$

де β – ставка обов'язкового єдиного внеску на державне соціальне страхування, %. В 2025 році ставка β = 22%. Тоді:

$$НЗН_{зп} = (33592 + 4703) \times 0,22 = 8424,90 \approx 8425 \text{ грн.}$$

г). Амортизація основних засобів А, які використовувались під час виконання магістерської кваліфікаційної роботи:

$$А = \frac{Ц \times Н_а}{100} \times \frac{Т}{12} \text{ грн}, \quad (5.4)$$

де Ц – загальна балансова вартість основних засобів, грн;

$Н_а$ – річна норма амортизаційних відрахувань.

Встановлено, що $Н_а = (2,5...25)\%$;

Т – термін використання основних засобів, місяці.

Зроблені розрахунки зведено в таблицю 5.5.

Таблиця 5.5 – Розрахунок амортизаційних відрахувань

Найменування обладнання, приміщень тощо	Балансова вартість, грн	Норма амортизації, %	Термін використання, місяців	Величина амортизаційних відрахувань, грн
1. Комп'ютерна техніка, обладнання, інші прилади тощо	100000	20,0	3,0 (при 60% використанні)	3000
2. Приміщення університету, факультету, кафедри	60000	2,5	3,0 (при 60% використанні)	225
3.Лазерна проєкційна система Multi GRAF1000	46000	25,0	3,0 (при 20% використанні)	575
4.Програмно-аналітичний комплекс	20000	25,0	3,0 (при 20% використанні)	250
Всього				А = 4050 грн

д). Витрати на матеріали М розраховуються за формулою:

$$M = \sum_{1}^{n} H_i \times Ц_i \times К_i - \sum_{1}^{n} B_i \times Ц_в \text{ грн}, \qquad (5.5)$$

де H_i – витрати матеріалу *і*-го найменування, кг;

$Ц_i$ – вартість матеріалу *і*-го най-менування;

$К_i$ – коефіцієнт транспортних витрат;

$К_i$ = (1,1…1,15); B_i – маса відходів матеріалу *і*-го найменування;

$Ц_в$ – ціна відходів матеріалу *і*-го найменування;

n – кількість видів матеріалів.

Під час виконання магістерської кваліфікаційної роботи загальні витрати на матеріали склали укрупнено приблизно 500 грн.

ж). Витрати на силову електроенергію $В_е$ розраховуються за формулою:

$$В_е = \frac{В \times П \times Ф \times К_п}{К_д}, \quad (5.6)$$

де В – вартість 1 кВт-год. електроенергії, в 2025 р. В ≈ 6,5 грн/кВт;

П – установлена потужність обладнання, кВт; П ≈ 1,00 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

Приймемо, що Ф = 235 годин;

$К_п$ – коефіцієнт використання потужності; $К_п < 1 = 0,73$.

$К_д$ – коефіцієнт корисної дії, $К_д = 0,62$.

Тоді витрати на силову електроенергію будуть дорівнювати:

$$В_е = \frac{В \times П \times Ф \times К_п}{К_д} = \frac{6,5 \times 1,00 \times 235 \times 0,73}{0,62} = 1798,51 \approx 1799 \text{ грн.}$$

и). Інші витрати $В_{інш}$ можна прийняти як (50…300)% від основної заробітної плати розробників, тобто:

$$В_{інш} = (0,5....3) \times З_о. \quad (5.7)$$

Для нашого випадку отримаємо:

$$В_{інш} = 0,5 \times 33592 = 16796 \text{ грн.}$$

к). Сума всіх попередніх статей витрат становить витрати на виконання цієї магістерської роботи безпосередньо розробником-здобувачем-магістрантом – В.

$$В = 33592 + 4703 + 8425 + 4050 + 500 + 1799 + 16796 = 69865 \text{ грн.}$$

л). Загальні витрати на завершення досліджень становитимуть:

$$В_{заг} = \frac{В}{b}, \qquad (5.8)$$

де b – коефіцієнт, який характеризує етап (стадію) виконання цієї роботи.

Оскільки наша розробка на цей момент часу потребує значного доопрацювання, то можна умовно прийняти, що, $b \approx 0{,}5$ [25].

Тоді: $В_{заг} = \frac{69865}{0{,}5} = 139730{,}00$ грн або приблизно 140 тисяч грн.

Тобто прогнозовані загальні витрати на проведення наукових досліджень побудови автоматизованих систем з підбору комп'ютерних комплектуючих можуть становити 140 тисяч грн.

5.3 Оцінювання наукового рівня проведених досліджень та можливості комерційного використання отриманих результатів

Оскільки завершення нашої роботи вимагає ще багато часу, то для кількісного оцінювання наукового рівня результатів проведених досліджень може бути використаний комплексний показник $К_р$, який розраховується за формулою [25]:

$$K_p = \frac{I^n \times T_c \times R}{B \times t}, \qquad (5.9)$$

де: I – коефіцієнт важливості досліджень, I = (2…5);

n – коефіцієнт використання результатів досліджень;

n = 0, коли результати роботи не будуть використовуватись;

n = 1, коли результати будуть використовуватись частково;

n = 2, коли результати роботи будуть використовуватись в дослідно-конструкторських розробках;

n = 3, коли результати можуть використовуватись навіть без проведення дослідно-конструкторських розробок;

T_c – коефіцієнт складності досліджень, $T_{скл}$ = (1…3);

R – коефіцієнт результативності досліджень:

R = 4, якщо результати роботи плануються вище відомих;

R = 3, якщо результати роботи відповідають відомому рівню;

R = 2, якщо результати нижче відомих;

R = 1, якщо результат роботи не визначений;

B – вартість (або витрати) проведених досліджень; для нашого випадку B = 140 тисяч грн;

t – час проведення подальших досліджень, років.

Якщо $K_p > 1$, то науковий рівень отриманих результатів проведених досліджень є високим.

При $K_p \approx 1{,}0$ науковий рівень отриманих результатів проведених досліджень є задовільним.

Для визначення коефіцієнтів, наведених у формулі 5.9, запросимо тих же експертів, які здійснювали технологічний аудит.

Результати висновків експертів занесено у таблицю 5.6.

Таблиця 5.6 – Результати оцінювання експертами зазначених коефіцієнтів

Показник	Експерти			Переважаюча (усереднена) оцінка
	Гармаш В.В.	Богач І.В.	Кулик Я.А.	
1. Коефіцієнт важливості роботи, I	4	4	3,5	4
2. Коефіцієнт використання результатів роботи, n	2,0	2,1	2	2
3. Коефіцієнт складності досліджень, T_c	1,2	2,7	3,0	2,5
4. Коефіцієнт результативності роботи, R	1,9	2,5	2,7	2,3
5. Вартість роботи, тисяч грн	140	140	140	140
6. Час проведення подальших досліджень, роки	0,5	0,4	0,5	0,5

Аналізуючи результати, наведені в таблиці 5.6, можна зробити висновок, що переважаючими (або усередненими) коефіцієнтами, які були виставлені експертами, будуть такі:

I – коефіцієнт важливості проведених досліджень, I = 4;

n – коефіцієнт використання результатів роботи; n = 2,0;

T_c – коефіцієнт складності досліджень, $T_{скл}$ = 2,5;

R – коефіцієнт результативності роботи; R = 2,5;

B – вартість роботи; B = 140 тисяч грн.

t – час завершення досліджень, t = 0,5 рік.

Тоді показник K_p, що визначає технічний рівень результатів, отриманих під час проведених досліджень, буде дорівнювати (табл. 5.7):

$$K_p = \frac{I^n \times T_c \times R}{B \times t} = \frac{4^2 \times 2,5 \times 2,3}{140 \times 0,5} = \frac{16 \times 2,5 \times 2,3}{140 \times 0,5} = 1,314 .$$

Таблиця 5.7 – Результати виконаної економічної частини магістерської кваліфікаційної роботи зведено у таблицю:

Показники	Задані у ТЗ	Досягнуто у магістерській кваліфікаційній роботі	Висновок
1. Витрати на виконання роботи	Не більше 150 тисяч грн	140 тисяч грн	Виконано
2. Коефіцієнт використання результатів проведених досліджень	не менше 2	2	Виконано
3. Коефіцієнт складності проведених досліджень	не менше 2,5	2,5	Виконано
4. Коефіцієнт результативності проведених досліджень	не менше 2,0	2,3	Виконано
5. Комплексний показник, що характеризує технічний рівень отриманих результатів	не менше 1,0	1,314	Досягнуто

5.4 Висновки до розділу 5

В результаті економічних розрахунків можна дійти до висновків, що оскільки K_p = 1,314 > 1, то це може свідчити про те, що науковий рівень результатів проведених досліджень та можливість комерціалізації отриманих результатів є високими.

ВИСНОВКИ

У даній роботі виконано комплексне дослідження, проєктування, розробку та тестування автоматизованої системи підбору комп'ютерних комплектуючих, що охоплює повний цикл створення функціональної веб-платформи електронної комерції.

У першому розділі проведено аналіз сучасних технологій та методів підбору комп'ютерних комплектуючих, де встановлено актуальність індивідуальної збірки персональних комп'ютерів як ефективної альтернативи готовим рішенням завдяки можливості повного контролю якості компонентів, адаптації до конкретних потреб та оптимізації бюджету. Визначено основні критерії якісного складання конфігурації, серед яких сумісність компонентів, енергетичний баланс, тепловий режим та відсутність вузьких місць у системі. Проаналізовано існуючі конфігуратори TELEMART та COMPX, що підтвердило доцільність використання онлайн-інструментів з автоматичною перевіркою сумісності для мінімізації помилок підбору. Виконано постановку задачі з визначенням функціональних та нефункціональних вимог до майбутньої системи, включаючи аутентифікацію користувачів, управління товарами, обробку замовлень, платіжну систему, адміністративний модуль та вимоги до безпеки, продуктивності й масштабованості.

У другому розділі здійснено комплексне моделювання архітектури системи засобами UML, що включає статичні та динамічні аспекти функціонування. Розроблено діаграму класів, яка визначає структуру системи через ключові сутності Product, ProductCategory, Order, OrderLineItem, UserProfile, Post, Replies, ForumCategory, ContactInfo та Newsletter з детальним описом атрибутів, методів та взаємозв'язків між класами, забезпечуючи модульність та масштабованість архітектури. Створено діаграму послідовності, що демонструє динамічну взаємодію компонентів під час процесу оформлення

замовлення через чотирнадцять послідовних кроків від перегляду профілю до відображення підтвердження, з чітким розмежуванням синхронних та асинхронних викликів між об'єктами. Спроєктовано структурну схему багаторівневої архітектури системи, організованої за принципом розділення відповідальностей на чотири рівні: презентації з веб-інтерфейсом, мобільним додатком, API Gateway, адмін-панеллю та email-клієнтом, бізнес-логіки з сервісами UserProfile, Product, Order, Post, Newsletter та Community, даних зі спеціалізованими базами даних для користувачів, товарів, замовлень, контенту, розсилок та аналітики, а також рівня зовнішніх сервісів та інфраструктури з платіжними шлюзами, хмарним сховищем, email-сервісами, CDN, аналітикою та компонентами безпеки й DevOps.

У третьому розділі виконано програмну реалізацію системи з обґрунтуванням вибору технологічного стека, включаючи Django як основний фреймворк, Python для бізнес-логіки, HTML/CSS/JavaScript для фронтенду, Bootstrap для адаптивного дизайну, PostgreSQL як базу даних, Stripe для обробки платежів, AWS S3 для зберігання медіа, Heroku для розгортання та додаткові інструменти Allauth, Gunicorn, Crispy Forms та інші, що забезпечило надійність, безпеку та масштабованість системи. Здійснено розробку основних модулів, зокрема webhook_handler.py для обробки вебхуків Stripe з методами handle_event, handle_payment_intent_succeeded та handle_payment_intent-_payment_failed, що забезпечують надійну обробку платежів з п'ятьма повторними спробами створення замовлення, автоматичним оновленням профілів та відправкою підтверджувальних email, а також views.py для реалізації функцій cache_checkout_data, checkout та checkout_success, які забезпечують повний цикл обробки корзини, створення замовлення, інтеграцію з платіжною системою та оновлення користувацьких даних з використанням динамічних шаблонів Django та валідації форм.

У четвертому розділі проведено комплексне тестування розробленої системи з демонстрацією основних інтерфейсів, включаючи головну сторінку з адаптивним хедером, сторінку реєстрації з валідацією email, каталог комплектуючих з фільтрацією за категоріями, алфавітом, рейтингом та цінами, детальні сторінки товарів з можливістю додавання в кошик, функціонал оплати через Stripe, систему wishlist для зареєстрованих користувачів та адміністративну панель для управління товарами. Виконано повне функціональне тестування системи за двадцятьма сімома тест-кейсами, що охоплюють створення облікових записів, аутентифікацію, навігацію, пошук та фільтрацію товарів, управління кошиком, процес оформлення замовлення з інтеграцією Stripe, відправку підтверджувальних email, роботу з wishlist, контактні форми, підписку на розсилку та адміністративні функції, причому всі тести успішно пройдено, що підтверджує повну працездатність та коректність розробленої системи. В результаті економічних розрахунків можна дійти до висновків, що оскільки $К_р = 1,314 > 1$, то це може свідчити про те, що науковий рівень результатів проведених досліджень та можливість комерціалізації отриманих результатів є високими. Також при розрахунку витрат виконано технічну умову не перевищувати суму 150 тисяч гривень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Барабан М. В., Гордієнко О. О. Автоматизована система з підбору комп’ютерних комплектуючих // Матеріали Міжнародної науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)» [Електронний ресурс]. URL: https ://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/viewFile/25750/21235 (дата звернення: 05.12.2025).

2. Шеховцов В. А. Операційні системи. Київ : Видавнича група BHV, 2005. 576 с.

3. Черняк О. І., Захарченко П. В. Інтелектуальний аналіз даних : підручник. Київ : Знання, 2014. 599 с.

4. PCMark 10 — The Complete Benchmark [Електронний ресурс]. URL: https ://benchmarks.ul.com/pcmark10 (дата звернення: 05.12.2018).

5. Лувішіс І., Зарубін Ю., Мазо Б. Коротко про історію розвитку комп’ютерів // Комп’ютер Прес. 1996. № 5.

6. Тхір І. Л., Калушка В. П., Юзьків А. В. Посібник користувача ПК. 3-тє вид. Тернопіль : Підручники та посібники, 2006. 1024 с.

7. 3DMark [Електронний ресурс]. URL: https ://uk.wikipedia.org/wiki/3DMark (дата звернення: 05.12.2025).

8. Мельник А. О. Архітектура комп’ютера: наук. вид. Луцьк: Волинська обласна друкарня, 2008. 470 с.

9. Шимон О. М. Апаратне та програмне забезпечення ПК : конспект лекцій. Житомир: ЖДУ ім. І. Франка, 2006. 17 с. URL: https ://eprints.zu.edu.ua/18/1/Konspect_modul_1_Windows.pdf (дата звернення: 05.12.2025).

10. Dynamic-link libraries [Electronic resource]. URL: https ://msdn.microsoft.com/en-us/library/windows/desktop/ms682589(v=vs.85).aspx (accessed: 05.12.2013).

11. Ситник В. Ф., Писаревська Т. А., Єрьоміна Н. В., Краєва О. С. Основи інформаційних систем : навч. посібник. Київ, 2005. 308 с.

12. Unified Modeling Language [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/Unified_Modeling_Language (дата звернення: 05.12.2025).

13. Microsoft Visual Studio [Електронний ресурс]. URL: https ://uk.wikipedia.org/wiki/Microsoft_Visual_Studio (дата звернення: 05.12.2025).

14. Josuttis N. M. The C++ Standard Library: A Tutorial and Reference. 2nd ed. Boston : Addison-Wesley, 2012. 1136 p.

15. Swaroop C. H. A Bite of Python. Version 2.01. 2013. 159 p.

16. PyCharm [Electronic resource]. URL: https://www.jetbrains.com/pycharm/ (accessed: 05.12.2025).

17. Chollet F. Deep Learning with Python. New York : Manning Publications, 2018. 400 p.

18. Grinberg M. Flask Web Development. Sebastopol : O’Reilly Media, 2018. 258 p.

19. Галовіц Я. C++17 STL. Стандартна бібліотека шаблонів. Київ : BHV, 2018. ISBN 978-5-4461-0680-6.

20. Парадигми програмування. Елементи програм C++ [Електронний ресурс]. URL: https ://om.univ.kiev.ua/users_upload/15/upload/file/prog_lecture_01.pdf (дата звернення: 05.12.2025).

21. Кравець П. О. Об’єктно-орієнтоване програмування: навч. посібник. Львів: Видавництво Львівської політехніки, 2012. 624 с.

22. C Tutorial [Electronic resource]. URL: https ://www.tutorialspoint.com/cprogramming/index.htm (accessed: 05.12.2025).

23. Офіційний сайт мови Python [Електронний ресурс]. URL: https ://www.python.org/ (дата звернення: 05.12.2025).

24. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спец. 126, 151, 174 / уклад. О. В. Бісікало та ін. Вінниця: ВНТУ, 2023. 64 с.

25. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / уклад. В. О. Козловський та ін. Вінниця: ВНТУ, 2021. 42 с.

ДОДАТКИ

Додаток А (обов'язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

ЗАТВЕРДЖУЮ
Завідувач кафедри АІІТ
проф., д.т.н. Олег БІСІКАЛО
“____”_____2025 року

ТЕХНІЧНЕ ЗАВДАННЯ

На магістерську кваліфікаційну роботу

«АВТОМАТИЗОВАНА СИСТЕМА З ПІДБОРУ КОМП’ЮТЕРНИХ КОМПЛЕКТУЮЧИХ»

08-31.МКР.002.02.000 ТЗ

Керівник роботи: к.т.н., доцент каф. АІІТ
Марія БАРАБАН
«16» жовтня 2025р.

Виконавець: ст.гр. 1АКІТР-24м
Олександр ГОРДІЄНКО
«16» жовтня 2025р.

Вінниця ВНТУ – 2025

1. Назва та галузь застосування

У даній роботі розглядається автоматизована система з підбору комп’ютерних комплектуючих, яка допоможе обрати потрібні компоненти для комп'ютерної техніки покупцям.

2. Підстави для розробки

Розробку системи здійснювали на підставі наказу по університету №313 від 24 вересня 2025 та завдання до магістерської кваліфікаційної роботи складеного та завтердженого кафедрою «Автоматизації та інтелектуальних інформаційних технологій».

3. Мета та призначення розробки

Метою роботи є удосконалення системи підбору комп’ютерних комплектуючих для підвищення ефективності та якості складання комп’ютерної техніки за рахунок розробки та впровадження автоматизованої системи, що забезпечить:

- автоматизований підбір комп’ютерних комплектуючих з узгодженням їх технічних характеристик;

- ефективну взаємодію з користувачами, які мають змогу придбати комп’ютерні комплектуючі;

- оптимізацію технічних параметри обраної комп’ютерної техніки.

Призначення системи: забезпечення автоматизованого підбору комп’ютерних комплектуючих з узгодженням їх технічних параметрів, сумісності та цінової складової.

4. Джерела розробки

1. Шеховцов В.А. Операційні системи . – К. : Видавнича група BHV, 2005. – 576с.

2. Інтелектуальний аналіз даних: Підручник / О.І. Черняк, П.В. Захарченко ; Київський національний університет ім. Т. Шевченка. — К. :

Знання, 2014. — 599 с. — (Університетський підручник).

5. Показники призначення

5.1 Основні технічні характеристики системи

Функціональні можливості:

5.1.1 Система аутентифікації та управління обліковими записами:

- має забезпечити реєстрацію користувачів з підтвердженням електронної пошти, авторизацію, вихід із системи, можливість відновлення пароля. Для авторизованих користувачів – доступ до профілю з редагуванням даних та перегляд історії замовлень;

5.1.2 Навігація сайтом:

- має реалізувати адаптивну навігаційну панель з логотипом, яка змінює прозорість при прокручуванні сторінки. Меню динамічно змінюється залежно від статусу користувача (гість, авторизований, адміністратор).

5.1.3 Управління товарами (каталог):

- має забезпечити перегляд списку товарів, відображення карток з фото, назвою, ціною, рейтингом, категорією. Реалізувати пошук, фільтрацію за категоріями та параметрами.

5.1.4 Сторінка товару:

- має відображати детальної інформації, наявності на складі, можливість додавання до кошика та список бажаного (wishlist), керування кількістю.

5.1.5 Кошик та оформлення замовлення:

- можливість перегляду доданих товарів, редагування кількості, видалення позицій. Під час оформлення – форма для введення контактних та платіжних даних із можливістю автозаповнення з профілю. Реалізувати систему онлайн-оплати через Stripe та підтвердження замовлення з надсиланням електронного листа.

5.1.6 Адміністративний модуль:

- адміністратор може додавати, редагувати та видаляти товари. Доступ – лише для авторизованих користувачів з роллю адміністратора. Перед видаленням – підтвердження через модальне вікно.

5.1.7 Контактна форма:

- має передбачити можливість надсилання звернень користувачами. Після заповнення – повернення на головну сторінку з відповідним повідомленням.

5.1.8 Динамічне оновлення кошика:

- при додаванні товару – автоматичне оновлення іконки кошика з показом суми та зміною кольору.

5.2. Мінімальні системні вимоги

5.2.1 Апаратне забезпечення:

- Процесор: тактова частота не менше 2.7 GHz, рекомендовано 4+ ядра;
- Оперативна пам'ять: 8 GB RAM;
- Місце на диску: 2 GB для програмного забезпечення та бібліотек; 5-10 GB для бази даних (залежить від кількості доданих товарів та користувачів);
- Мережа: стабільне Інтернет-з'єднання для завантаження даних (мінімум 10 Mbps).

5.2.2 Програмне забезпечення:

- Операційна система: Windows 10/11;
- наявність бібліотеки DirectX.

5.3. Вхідні дані

- кількість користувачів – до 100чол;
- кількість комплектуючих для аналізу – не менше 10шт;
- кількість одночасно підключених користувачів – не менше 50 чол.

5.4 Результати роботи програми

5.4.1 зручний GUI інтерфейс

5.4.2 звітна форма для перегляду покупок;

5.4.3 форма для зворотнього зв’язку з продавцем.

6. Економічні показники:

До економічних показників входять:

- Витрати на розробку – 140 тисяч грн;
- Комплексний показник, що характеризує технічний рівень отриманих результатів – 1,314.

Стадії розробки

Розділ 1 має бути виконаний до 05.10.2025 р.

Розділ 2 має бути виконаний до 15.10.2025 р.

Розділ 3 має бути виконаний до 31.10.2025 р.

Розділ 4 має бути виконаний до 10.11.2025 р.

Економічний розділ має бути виконаний до 01.12.2025 р.

8. Порядок контролю та приймання

Рубіжний контроль провести до 28.11.2025 р.

Попередній захист магістерської кваліфікаційної роботи провести до 03.12.2025р

Захист магістерської кваліфікаційної роботи провести до 19.12.2025 р.

Додаток Б (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

АВТОМАТИЗОВАНА СИСТЕМА З ПІДБОРУ КОМП'ЮТЕРНИХ КОМПЛЕКТУЮЧИХ

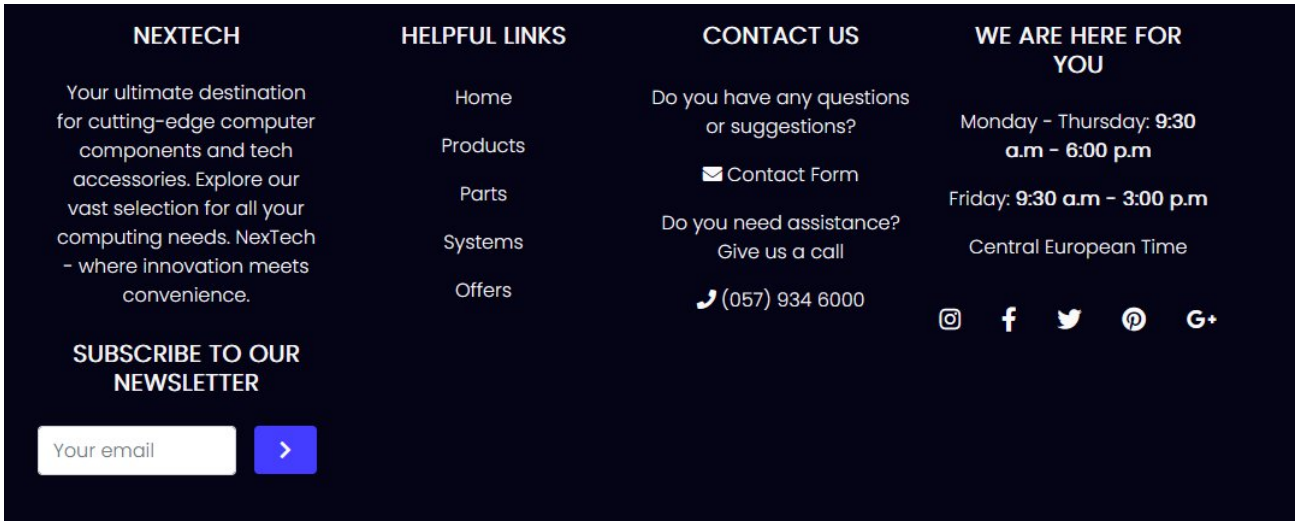


Рисунок Б.1 – Хедер головної сторінки

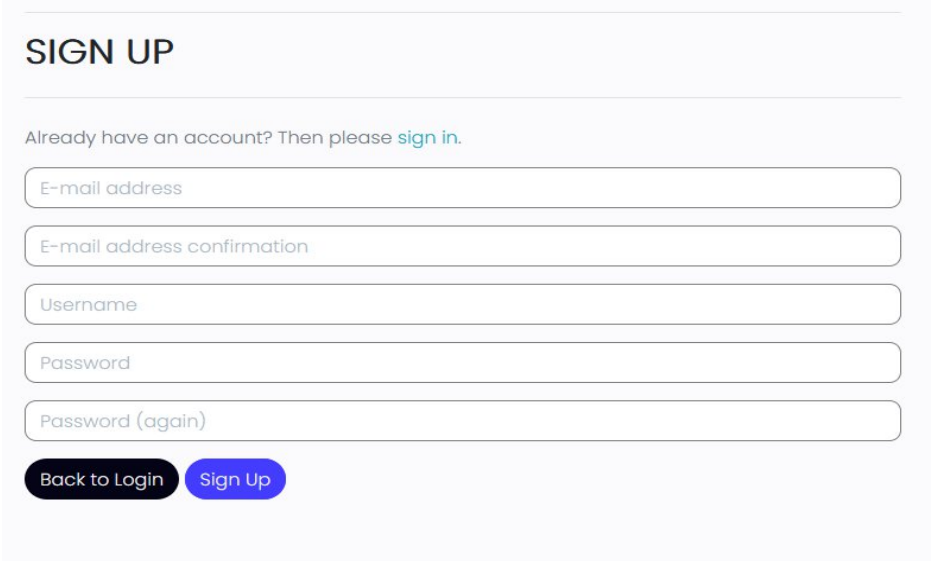


Рисунок Б.2 – Сторінка реєстрації в системі

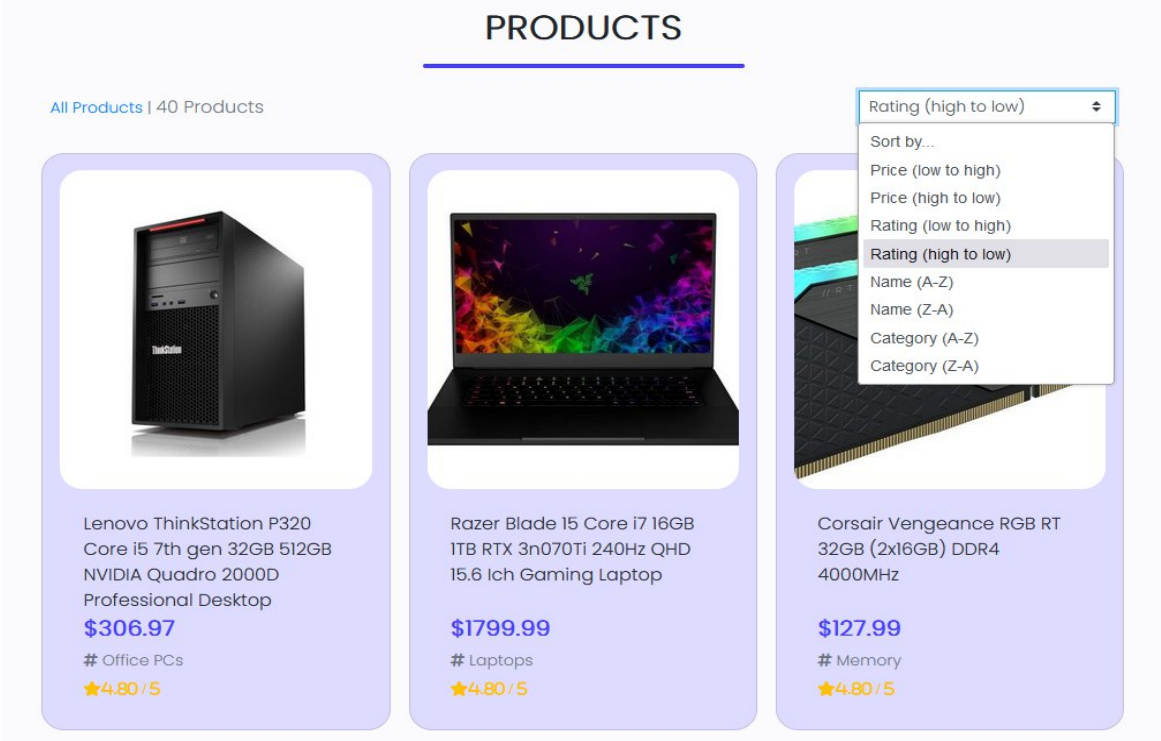


Рисунок Б.3 – Основна сторінка комплектуючих з фільтрами

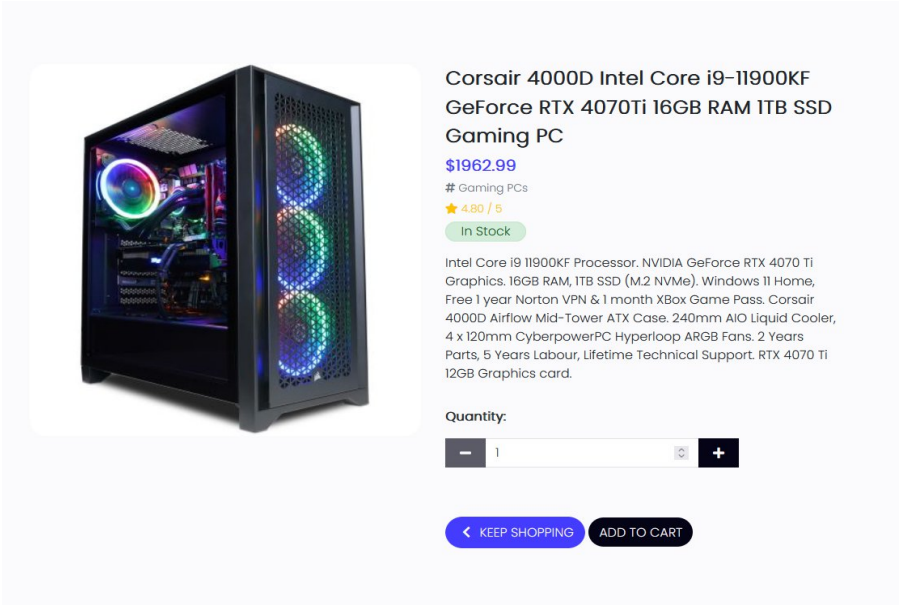


Рисунок Б.4 – Сторінка окремого комплектуючого компонента

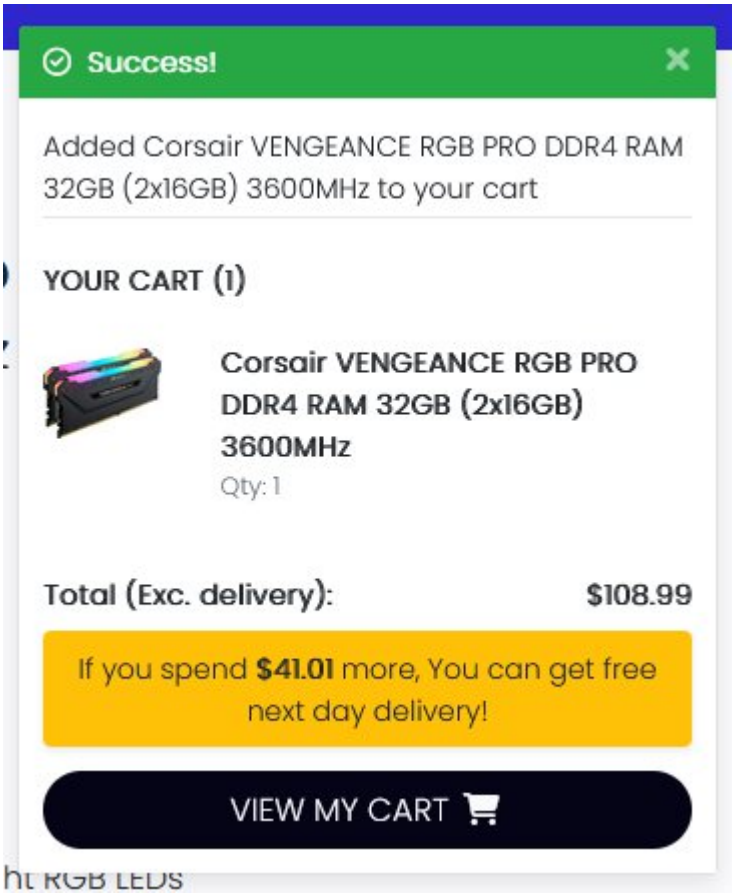


Рисунок Б.5 – Оплата замовлення

Рисунок Б.6 – Додавання товарів до вішліста

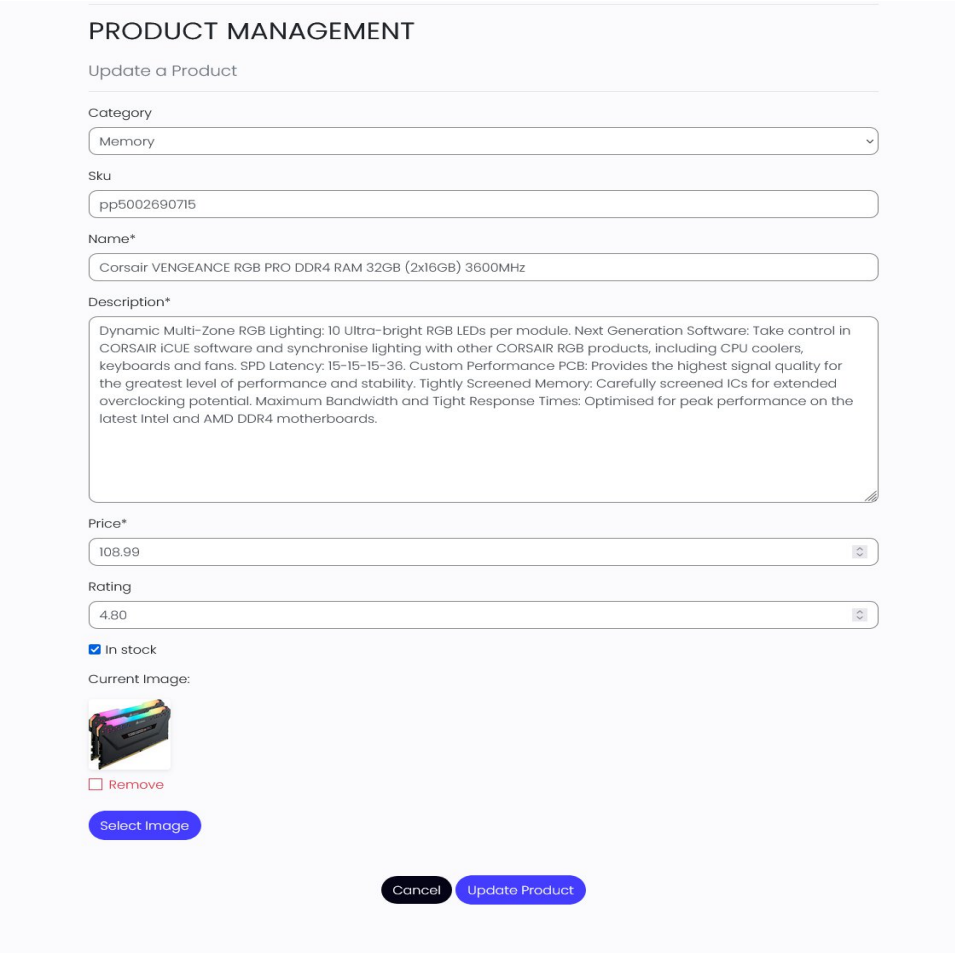


Рисунок Б.7 – Редагування опису товарів адміном

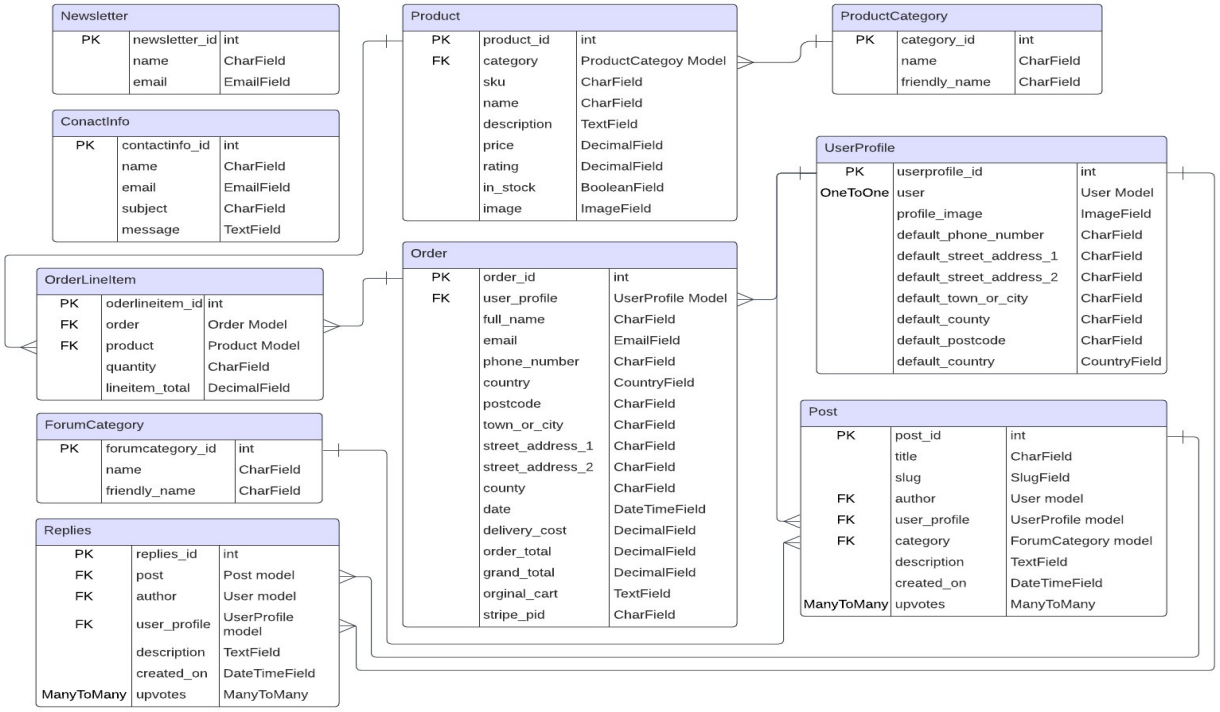


Рисунок Б.8 –Діаграма класів автоматизованої системи з підбору комп’ютерних комплектуючих

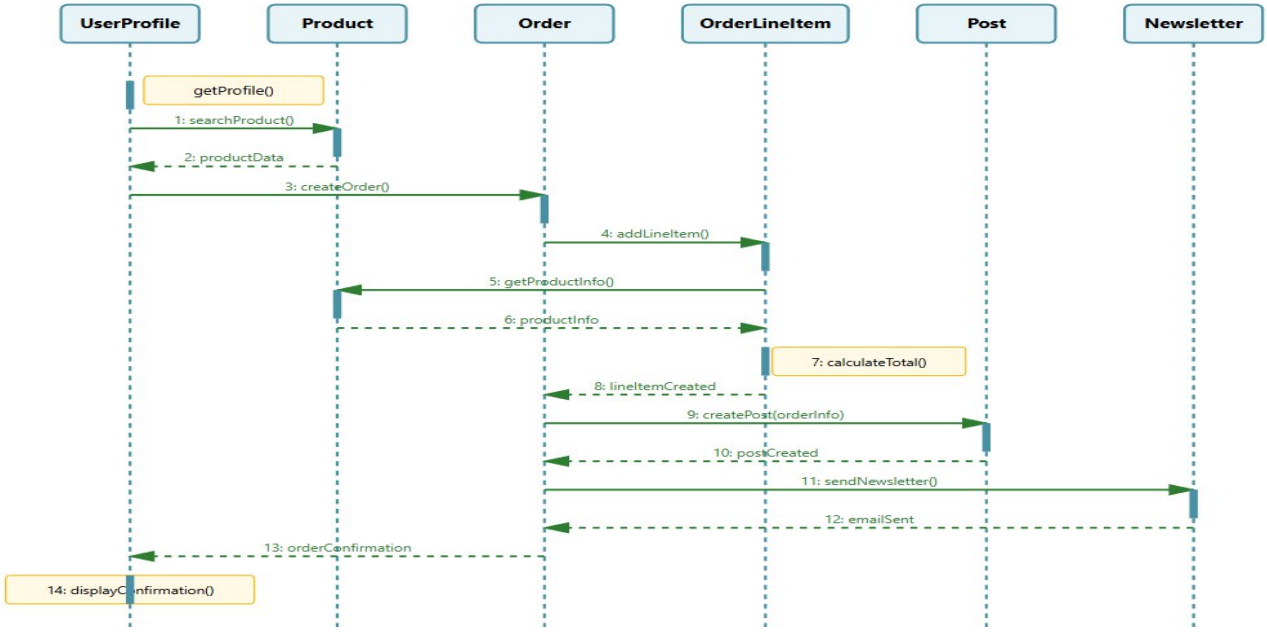


Рисунок Б.9 – Діаграма послідовності автоматизованої системи з підбору комп’ютерних комплектуючих

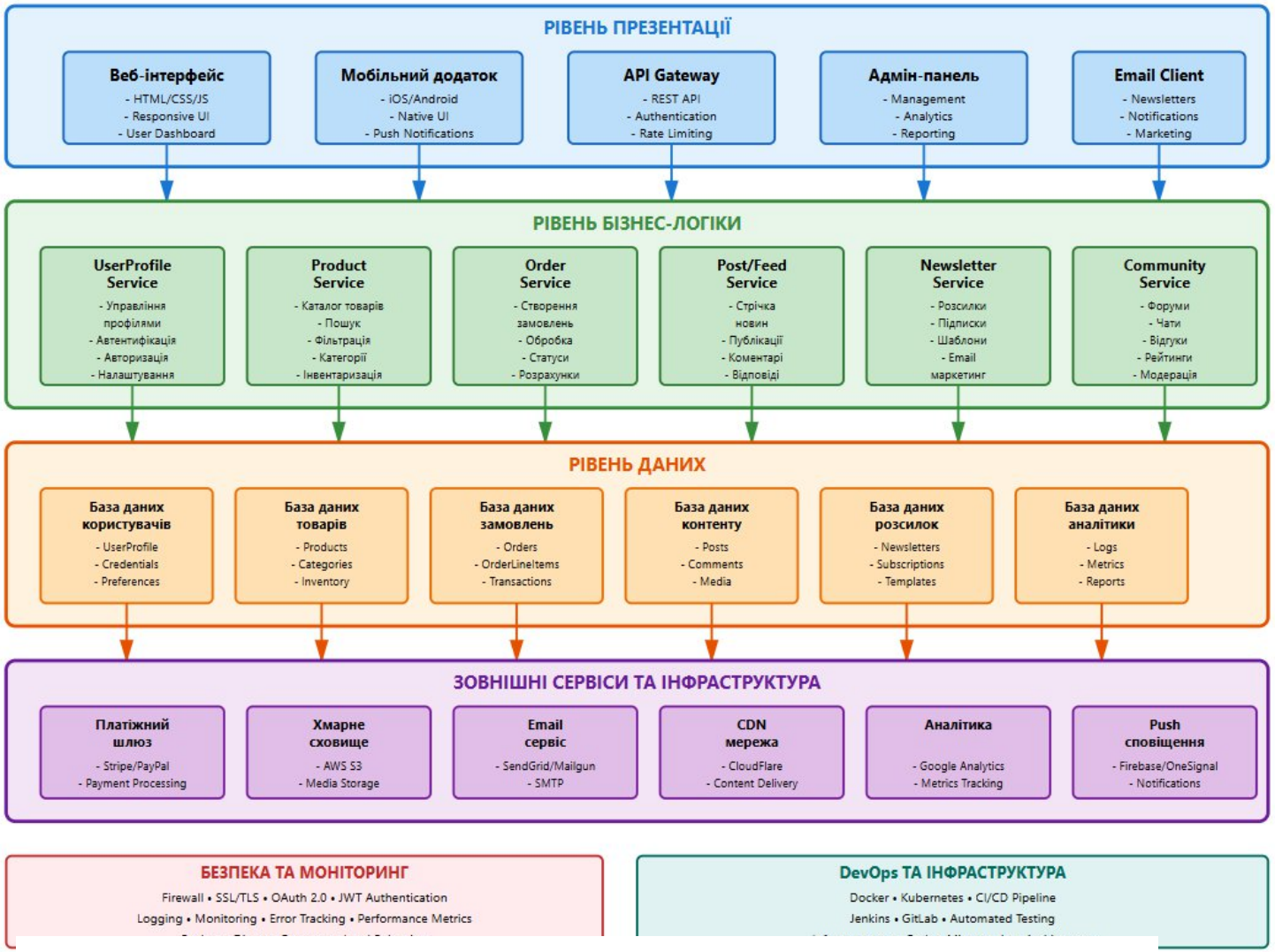


Рисунок Б.10 – Структурна схема автоматизованої системи з підбору комп’ютерних комплектуючих

Додаток В (обов’язковий)

ЛІСТИНГ ПРОГРАМИ

```
"""
Django settings for nextech project.

Generated by 'django-admin startproject' using Django 3.2.23.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""

from pathlib import Path
import os
import dj_database_url

if os.path.isfile('env.py'):
    import env
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent



# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = os.environ.get('SECRET_KEY')

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = 'DEVELOPMENT' in os.environ

if 'DEVELOPMENT' in os.environ:
    ALLOWED_HOSTS = [os.environ.get('LOCAL_HOSTNAME')]
else:
    ALLOWED_HOSTS = [os.environ.get('HEROKU_HOSTNAME')]

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.sites',
    'allauth',
    'allauth.account',
    'allauth.socialaccount',
    'home',
    'products',
```

```
    'cart',
    'checkout',
    'profiles',
    'contact',
    'wishlist',

    # Extra
    'crispy_forms',
    'storages',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'nextech.urls'

CRISPY_TEMPLATE_PACK = 'bootstrap4'

TEMPLATES = [
    {
```

```
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            os.path.join(BASE_DIR, 'templates'),
            os.path.join(BASE_DIR, 'templates', 'allauth'),
        ],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                # required by allauth
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'cart.contexts.cart_contents',
                'django.template.context_processors.media',
            ],
            'builtins': [
                'crispy_forms.templatetags.crispy_forms_tags',
                'crispy_forms.templatetags.crispy_forms_field',
            ]
        },
    },
]

# Store messages in the session
MESSAGE_STORAGE = 'django.contrib.messages.storage.session.SessionStorage'
```

```
AUTHENTICATION_BACKENDS = (
    # Needed to login by username in Django admin, regardless of `allauth`
    'django.contrib.auth.backends.ModelBackend',

    # `allauth` specific authentication methods, such as login by e-mail
    'allauth.account.auth_backends.AuthenticationBackend',
)

SITE_ID = 1

# Allows authentication using usernames OR emails
ACCOUNT_AUTHENTICATION_METHOD = 'username_email'
ACCOUNT_EMAIL_REQUIRED = True
ACCOUNT_EMAIL_VERIFICATION = 'mandatory'
ACCOUNT_SIGNUP_EMAIL_ENTER_TWICE = True
ACCOUNT_USERNAME_MIN_LENGTH = 4
LOGIN_URL = '/accounts/login/'
LOGIN_REDIRECT_URL = '/'

WSGI_APPLICATION = 'nextech.wsgi.application'



# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

if 'DATABASE_URL' in os.environ:
    DATABASES = {
```

```
        'default': dj_database_url.parse(os.environ.get('DATABASE_URL'))
    }
else:
    DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.sqlite3',
            'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
        }
    }
# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators


AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
```

```
]

# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

# Specify static files location
STATIC_URL = '/static/'
STATICFILES_DIRS = (os.path.join(BASE_DIR, 'static'),)

# specify media files location
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

# Default primary key field type
```

```
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

if 'USE_AWS' in os.environ:
    # Cache control
    AWS_S3_OBJECT_PARAMETERS = {
        'Expires': 'Thu, 31 Dec 2099 20:00:00 GMT',
        'CacheControl': 'max-age=94608000',
    }

    # Bucket Config
    AWS_STORAGE_BUCKET_NAME = 'nextech-md'
    AWS_S3_REGION_NAME = 'eu-west-1'
    AWS_ACCESS_KEY_ID = os.environ.get('AWS_ACCESS_KEY_ID')
    AWS_SECRET_ACCESS_KEY =
os.environ.get('AWS_SECRET_ACCESS_KEY')
    AWS_S3_CUSTOM_DOMAIN =
f'{AWS_STORAGE_BUCKET_NAME}.s3.amazonaws.com'

    # Static and media files
    STATICFILES_STORAGE = 'custom_storages.StaticStorage'
    STATICFILES_LOCATION = 'static'
    DEFAULT_FILE_STORAGE = 'custom_storages.MediaStorage'
    MEDIAFILES_LOCATION = 'media'

    # Override static and media URLs in production
```

```
    STATIC_URL =
f'https://{AWS_S3_CUSTOM_DOMAIN}/{STATICFILES_LOCATION}/'
    MEDIA_URL =
f'https://{AWS_S3_CUSTOM_DOMAIN}/{MEDIAFILES_LOCATION}/'


# Stripe
# Variables used to calculate delivery costs
FREE_DELIVERY_THRESHOLD = 150
STANDARD_DELIVERY_PERCENTAGE = 10
STRIPE_CURRENCY = 'usd'
# Stripe Keys
STRIPE_PUBLIC_KEY = os.getenv('STRIPE_PUBLIC_KEY', '')
STRIPE_SECRET_KEY = os.getenv('STRIPE_SECRET_KEY', '')
STRIPE_WH_SECRET = os.getenv('STRIPE_WH_SECRET', '')


# Email Config
if 'DEVELOPMENT' in os.environ:
    EMAIL_BACKEND = 'django.core.mail.backends.console.EmailBackend'
    DEFAULT_FROM_EMAIL = 'noreply@nextech.com'
else:
    EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
    EMAIL_USE_TLS = True
    EMAIL_PORT = 587
    EMAIL_HOST = 'smtp.gmail.com'
    EMAIL_HOST_USER = os.environ.get('EMAIL_HOST_USER')
    EMAIL_HOST_PASSWORD = os.environ.get('EMAIL_HOST_PASS')
    DEFAULT_FROM_EMAIL = os.environ.get('EMAIL_HOST_USER')
```

```
    from django.shortcuts import render, redirect, reverse, get_object_or_404
from django.contrib import messages
from django.contrib.auth.decorators import login_required
from django.db.models import Q
from django.db.models.functions import Lower

from .models import Product, ProductCategory
from .forms import ProductForm


def all_products(request):
    """ This view shows all products, including sorting and search queries """

    products = Product.objects.all()
    query = None
    categories = None
    sort = None
    direction = None

    if request.GET:
        # Sort products by price, rating or category
        if 'sort' in request.GET:
            sortkey = request.GET['sort']
            sort = sortkey
            if sortkey == 'name':
                sortkey = 'lower_name'
                products = products.annotate(lower_name=Lower('name'))
```

```
    if sortkey == 'category':
        sortkey = 'category__name'
    if 'direction' in request.GET:
        direction = request.GET['direction']
        if direction == 'desc':
            sortkey = f'-{sortkey}'
    products = products.order_by(sortkey)

# Return producs matching selected category
if 'category' in request.GET:
    categories = request.GET['category'].split(',')
    products = products.filter(category__name__in=categories)
    categories = ProductCategory.objects.filter(name__in=categories)

# Return products whos name or description match the search query
if 'q' in request.GET:
    query = request.GET['q']
    if not query:
        messages.error(
            request,
            "Please enter a keyword or phrase to initiate the search!"
        )
        return redirect(reverse('products'))

    # Allowing queries to be filtered based on name OR description
    queries = (
        Q(name__icontains=query) | Q(description__icontains=query))
```

```
            products = products.filter(queries)

    # Return current sorting methodology to the template
    current_sorting = f'{sort}_{direction}'

    context = {
        'products': products,
        'search_term': query,
        'current_categories': categories,
        'current_sorting': current_sorting,
    }

    return render(request, 'products/products.html', context)


def product_detail(request, product_id):
    """ This view shows each individual products details """

    product = get_object_or_404(Product, pk=product_id)

    context = {
        'product': product,
    }

    return render(request, 'products/product_detail.html', context)
```

```
@login_required
def add_product(request):
    """ This view handles adding a product to the store """
    # Prevents standard users from accessing view
    if not request.user.is_superuser:
        messages.error(request, 'Sorry, only verified users can do that.')
        return redirect(reverse('home'))

    if request.method == 'POST':
        form = ProductForm(request.POST, request.FILES)
        if form.is_valid():
            product = form.save()
            messages.success(request, 'Successfully added product!')
            return redirect(reverse('product_detail', args=[product.id]))
        else:
            messages.error(
                request,
                'Failed to add product. Please ensure the form is valid.'
                )
    else:
        form = ProductForm()

    template = 'products/add_product.html'
    context = {
        'form': form,
    }
```

```
    return render(request, template, context)


@login_required
def edit_product(request, product_id):
    """ This view handles editing existing products in the store """
    if not request.user.is_superuser:
        messages.error(request, 'Sorry, only verified users can do that.')
        return redirect(reverse('home'))

    # Pre fill form with existing data
    product = get_object_or_404(Product, pk=product_id)
    if request.method == 'POST':
        form = ProductForm(request.POST, request.FILES, instance=product)
        if form.is_valid():
            form.save()
            messages.success(request, 'Successfully updated product!')
            return redirect(reverse('product_detail', args=[product.id]))
        else:
            messages.error(
                request,
                'Failed to update product. Please ensure the form is valid.'
                )
    else:
        form = ProductForm(instance=product)
        messages.info(request, f'You are editing {product.name}')
```

```
    template = 'products/edit_product.html'
    context = {
        'form': form,
        'product': product,
    }

    return render(request, template, context)


@login_required
def delete_product(request, product_id):
    """ This view handles deleting products from the store """
    if not request.user.is_superuser:
        messages.error(request, 'Sorry, only verified users can do that.')
        return redirect(reverse('home'))

    product = get_object_or_404(Product, pk=product_id)
    product.delete()
    messages.success(request, f'{product.name} - successfully deleted!')
    return redirect(reverse('products'))
```

119

Додаток Г (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: «Автоматизована система з підбору комп'ютерних комплектуючих»

Тип роботи: магістерська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра АІІТ
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) 0,51 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ⦿ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.
- • У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- • У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Бісікало О.В., зав. каф. АІІТ (підпис)
(прізвище, ініціали, посада)

Овчинников К.В., доц. каф. АІІТ (підпис)
(прізвище, ініціали, посада)

Особа, відповідальна за перевірку (підпис) Маслій Р.В.
(прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник (підпис) Барабан М.В., доц. каф. АІІТ
(прізвище, ініціали, посада)

Здобувач (підпис) Гордієнко О.О.
(прізвище, ініціали)