МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_  _ проф. Приходько С.Б.

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНАРОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: ***Розробка вебзастосунку для автоматизації обробки онлайн-замовлень та підбору відеокарт для магазину «V-Core»***

Виконав: студент групи 4151

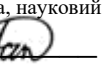
 Паученко П. О.

(підпис, ПІБ)

Керівник роботи:

Доцент кафедри ПЗАС, к. ф.-м. н., доцент

(посада, науковий ступень вчене звання)

 Латанська Л. О.

(підпис, ПІБ)

Миколаїв – 2026 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«_07_» ___04___ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Паученку Павлу Олеговичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка вебзастосунку для автоматизації обробки онлайн-замовлень та підбору відеокарт для магазину «V-Core»

Керівник роботи Латанська Людмила Олексіївна

Затверджені наказом ректора №275-уч від 07.04.2026 р.

2. Термін подання роботи: 01.06.2026 р.

3. Вихідні дані по роботі: _____

4. Перелік питань, що належать до розробки (найменування розділів)
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів 1 Титульний слайд, 2 Мета та завдання кваліфікаційної роботи, 3 Актуальність, аналіз існуючих рішень та постановка задачі, 4 Аналіз вимог (діаграма варіантів використання), 5 Архітектура ПЗ: клієнт-серверна структура (діаграма компонентів), 6 Архітектура ПЗ: фізичне розміщення (діаграма розгортання), 7 Ескізний проєкт: алгоритм підбору відеокарти (діаграма діяльності), 8 Технічний проєкт: статична структура вебзастосунку (діаграма класів), 9 Технічний проєкт: проєктування бази даних (концептуальна та фізична моделі), 10 Робочий проєкт: вибір технологічного стеку (React, Node.js, SQLite), 11 Результати розробки: інтерфейс каталогу та багатофакторна фільтрація, 12 Результати розробки: модуль підбору відеокарт за параметрами системи, 13 Результати розробки: кошик та процес оформлення замовлення, 14 Результати розробки: панель адміністратора (управління асортиментом), 15 Результати розробки: модерація відгуків користувачів, 16 Аналіз функціонування та тестування вебсистеми, 17 Аналіз факторів та заходи з охорони праці, 18 Підготовлений комплект супровідної документації, 19 Висновки, 20 Заключний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 07.04.2026 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	30.03.2026	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	06.04.2026	ПП*
3.	Аналіз вимог до ПЗ	13.04.2026	ПП*
4.	Розробка архітектури ПЗ	20.04.2026	
5.	Розробка проєкту ПЗ	04.05.2026	
6.	Реалізація та тестування ПЗ	11.05.2026	
7.	Підготовка розділу Результати розробки ПЗ	15.05.2026	
8.	Підготовка розділу з охорони праці	18.05.2026	ПП*
9.	Підготовка розділу ВИСНОВКИ	22.05.2026	
10.	Оформлення списку використаних джерел та додатків	25.05.2026	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	01.06.2026	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку
12.	Підготовка презентації та доповіді	05.06.2026	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	08.06.2026	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	15.06.2026	

* - за результатами переддипломної практики (ПП), яка була з 02.02.2026 до 22.03.2026 р.

Студент _____ Паученко П. О.
(підпис) (ПІБ)

Керівник роботи _____ Латанська Л. О.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена дослідженню, проєктуванню та розробці вебсистеми для автоматизації онлайн-замовлень та підбору відеокарт магазину «V-Core». Ця задача є типовою практичною задачею інженерії програмного забезпечення, яка характеризується високим рівнем комплексності та спрямована на оптимізацію бізнес-процесів роздрібної торгівлі комп'ютерними комплектуючими.

У процесі роботи було проведено детальне дослідження предметної галузі та порівняльний аналіз існуючих програмних рішень (зокрема, аналогів на кшталт PCPartPicker та OpenCart), на основі якого сформульовано вимоги та розроблено технічне завдання. Спроєктовано та реалізовано програмну систему з використанням сучасного стеку вебтехнологій: бібліотеки React, мови програмування TypeScript та інструменту збірки Vite. Для розробки адаптивного та інтерактивного користувацького інтерфейсу застосовано CSS-фреймворк Tailwind CSS, бібліотеку анімацій Framer Motion та графічні компоненти Lucide React. Система має модульну архітектуру, що забезпечує виконання CRUD-операцій, ефективне управління складськими залишками, обробку замовлень та надання технічних рекомендацій користувачам щодо вибору апаратного забезпечення.

Проведено тестування розробленого вебзастосунку, що підтвердило коректність виконання ключових операцій та стабільність роботи системи. Результати показали, що впровадження розробленого програмного забезпечення дозволяє значно автоматизувати рутинні процеси онлайн-торгівлі. Також у роботі розглянуто аспекти охорони праці при розробці програмного забезпечення та проаналізовано структуру управління безпекою на робочому місці.

Кваліфікаційна робота викладена на 155 сторінках друкованого тексту, містить 26 рисунків, 9 таблиць, список використаних джерел з 11 найменувань та 6 додатків.

ABSTRACT

The qualification work is dedicated to the research, design, and development of a web system for the automation of online orders and video card selection for the "V-Core" shop. This problem is a typical practical task of software engineering, characterized by a high level of complexity and aimed at optimizing the business processes of computer components retail.

During the work, a detailed study of the subject area and a comparative analysis of existing software solutions (in particular, analogues such as PCPartPicker and OpenCart) were conducted, based on which requirements were formulated and technical specifications were developed. The software system was designed and implemented using a modern web technology stack: the React library, the TypeScript programming language, and the Vite build tool. To develop an adaptive and interactive user interface, the Tailwind CSS framework, the Framer Motion animation library, and Lucide React graphic components were applied. The system has a modular architecture that ensures the execution of CRUD operations, efficient inventory management, order processing, and the provision of technical recommendations to users regarding hardware selection.

Testing of the developed web application was conducted, confirming the correct execution of key operations and system stability. The results showed that the implementation of the developed software allows for significant automation of routine e-commerce processes. The work also considers occupational safety aspects during software development and analyzes the occupational health and safety management structure in the workplace.

The qualification work consists of 155 pages of printed text, contains 26 figures, 9 tables, a list of references consisting of 11 sources, and 6 appendices.

ЗМІСТ

ВСТУП

В умовах сучасного інформаційного суспільства та стрімкого розвитку електронної комерції ефективність роздрібної торгівлі комп'ютерними комплектуючими безпосередньо залежить від якості та функціональності використовуваного програмного забезпечення [1]. Традиційні методи управління продажами вже не відповідають вимогам ринку, що зумовлює потребу у впровадженні інноваційних вебрішень для підвищення ефективності бізнес-процесів. Зокрема, сучасні вебсистеми у сфері торгівлі комп'ютерною технікою вимагають надійної основи для зберігання та оперативної обробки великих масивів даних зі складними зв’язками між технічними характеристиками товарів.

Ринок комп'ютерної техніки відзначається високою волатильністю та швидкою зміною поколінь пристроїв. При виборі комплектуючих, таких як відеокарти, ціна помилки є високою, тому програмне забезпечення має забезпечувати не лише базовий продаж, а й автоматизовану допомогу у технічно обґрунтованому підборі товару. Відсутність спеціалізованого модуля перевірки технічної сумісності (наприклад, відповідність TDP відеокарти можливостям блоку живлення або архітектурні обмеження інтерфейсів) призводить до підвищення навантаження на персонал та зростання кількості помилкових замовлень. Існуючі рішення або є занадто загальними, або не дозволяють реалізувати специфічну технічну логіку без суттєвих витрат.

Дана кваліфікаційна робота присвячена розв'язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці спеціалізованого вебзастосунку для автоматизації ключових операцій магазину «V-Core», що спеціалізується на реалізації відеокарт. Створення індивідуального програмного рішення дозволить автоматизувати внутрішні процеси, об'єднавши каталог, систему замовлень та

облік залишків у єдиний інструмент. Це забезпечить точність вибору завдяки фільтрам за критичними параметрами та оптимізує користувацький досвід.

Метою кваліфікаційної роботи є проєктування, реалізація та тестування сучасного вебзастосунку, який поєднує в собі функціонал потужної платформи електронної комерції та системи для технічного підбору відеокарт для магазину «V-Core».

Для досягнення поставленої мети необхідно:

– проаналізувати практичну задачу інженерії програмного забезпечення щодо автоматизації роботи магазину «V-Core»;

– проаналізувати існуючі аналоги, виявити їхні переваги та недоліки, і на основі цього обґрунтувати доцільність розробки власного вебзастосунку;

– сформулювати постановку задачі на розробку вебзастосунку для магазину «V-Core»;

– провести аналіз вимог до вебзастосунку, розробивши модель варіантів використання та специфікації прецедентів;

– розробити архітектуру вебзастосунку для магазину «V-Core»;

– розробити проєкт вебзастосунку, послідовно виконавши ескізне, технічне та робоче проєктування;

– здійснити програмну реалізацію, тестування та представити кінцеві результати розробки вебзастосунку;

– опрацювати питання охорони праці при роботі на комп'ютерному обладнанні;

– розробити необхідну програмну та експлуатаційну документацію, включаючи технічне завдання, програму і методику випробувань та інструкцію користувача.

Об’єктом кваліфікаційної роботи є процес розробки програмного забезпечення для автоматизації бізнес-процесів роздрібної торгівлі комп’ютерними комплектуючими у магазині «V-Core».

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ ОБЛІКУ ОНЛАЙН-ЗАМОВЛЕНЬ ТА ПІДБОРУ ВІДЕОКАРТ МАГАЗИНУ «V-CORE»

1.1 Аналіз практичної задачі інженерії програмного забезпечення для автоматизації бізнес-процесів магазину «V-Core»

Магазин «V-Core» функціонує на ринку комп'ютерних комплектуючих, пропонуючи товари для створення та модернізації персональних комп'ютерів. Основною спеціалізацією магазину є роздрібна реалізація відеокарт різних цінових сегментів та архітектурних поколінь. Основна діяльність магазину охоплює:

– Продаж комп'ютерних комплектуючих**.** У «V-Core» представлено продукцію провідних світових виробників графічних процесорів (NVIDIA, AMD, Intel) та різноманітних вендорів. Це забезпечує широкий вибір для вирішення різних завдань: від базових офісних потреб до складання високопродуктивних ігрових та професійних робочих станцій.

– Консультації та технічну підтримку клієнтів. Кваліфіковані працівники надають рекомендації щодо вибору оптимальної відеокарти, детально аналізуючи наявну конфігурацію ПК клієнта. Враховуються такі критичні параметри, як потужність блоку живлення (TDP), форм-фактор корпусу, сумісність інтерфейсів материнської плати та процесора. Такий підхід допомагає мінімізувати ризики придбання несумісного обладнання.

– Управління асортиментом та складський облік. Магазин взаємодіє з постачальниками та веде облік наявних на складі товарів, реагуючи на постійні зміни в асортименті та коливання ринкових цін на комп'ютерну техніку.

На даний момент магазин «V-Core» здійснює свою діяльність переважно через пряме спілкування з клієнтами (офлайн-продажі, телефонні консультації або замовлення через месенджери). Магазин не має власної вебплатформи чи інтернет-магазину, що суттєво обмежує його аудиторію, географію продажів та ускладнює процес самостійного ознайомлення клієнтів з наявним асортиментом і технічними характеристиками товарів.

Без спеціалізованого програмного забезпечення та автоматизованої вітрини менеджери змушені вручну вести облік залишків, приймати замовлення та фіксувати фінансові операції у розрізнених електронних таблицях або зошитах. Відсутність автоматизованого інструменту для підбору комплектуючих призводить до того, що перевірка технічної сумісності лягає виключно на персонал. Зважаючи на велику кількість технічних параметрів сучасних відеокарт (обсяг і тип відеопам'яті, шина, необхідні роз'єми додаткового живлення тощо), ручна перевірка значно збільшує час обслуговування кожного клієнта. Постійне навантаження підвищує ризик людської помилки під час консультацій, що може призвести до повернень товару або незадоволеності клієнтів.

Внаслідок таких недоліків загальна ефективність бізнес-процесів знижується, а магазин втрачає значну частину потенційних покупців, які віддають перевагу зручним онлайн-покупкам. З огляду на ці фактори, виникає потреба в автоматизації вказаних бізнес-процесів.

1.2 Аналіз існуючих програмних рішень

Розглянемо вже існуючі програмні рішення. Одне з таких – глобальний сервіс підбору та планування конфігурацій персональних комп’ютерів PCPartPicker. Це світовий лідер у своїй сфері, який володіє величезною базою даних технічних характеристик комплектуючих та дозволяє перевіряти їх сумісність між собою [2].

Головна сторінка сервісу PCPartPicker зображена на рисунку 1.1

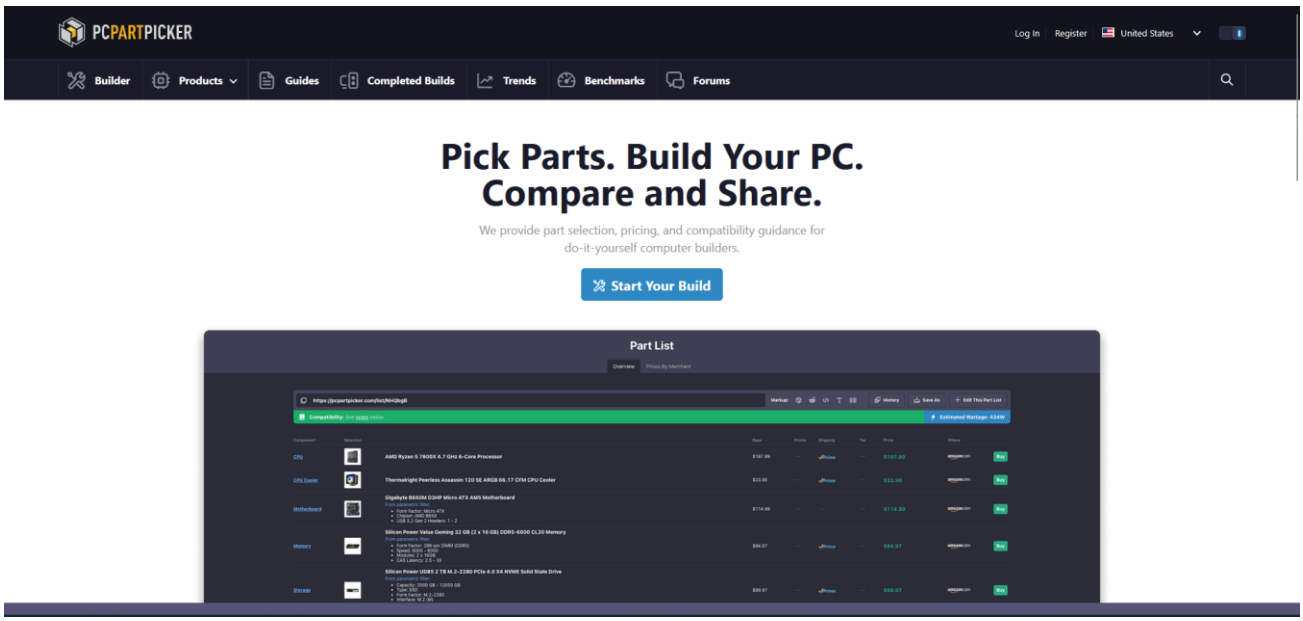


Рисунок 1.1 – Головна сторінка сервісу PCPartPicker

Переваги:

– наявність потужних алгоритмів перевірки технічної сумісності обладнання;

– автоматичне порівняння цін у різних світових магазинах;

– велика та активна спільнота користувачів, зручний візуальний інтерфейс.

Недоліки:

– це сторонній агрегатор, а не система управління конкретним магазином;

– сервіс не дозволяє автоматизувати внутрішні бізнес-процеси магазину «V-Core»;

– не підтримує локальний облік залишків на складі та не має функціоналу для обробки замовлень адміністраторами.

Таким чином, хоча PCPartPicker демонструє високий рівень автоматичного підбору комплектуючих, це закритий глобальний агрегатор, який неможливо використовувати для управління внутрішніми процесами магазину. Для бізнесу «V-Core» потрібне рішення, яке дозволить не лише консультувати клієнтів, а й повноцінно обробляти власні замовлення та вести локальний складський облік.

Ще один аналог програмного забезпечення – спеціалізований інтернет-магазин комп'ютерної техніки з функцією підбору комплектуючих, на прикладі вітчизняної платформи Telemart [3]. Це один із лідерів ринку, який спеціалізується саме на комп'ютерних комплектуючих та пропонує користувачам інтерактивний інструмент «Збірка ПК».

Головна сторінка інтернет-магазину Telemart зображена на рисунку 1.2

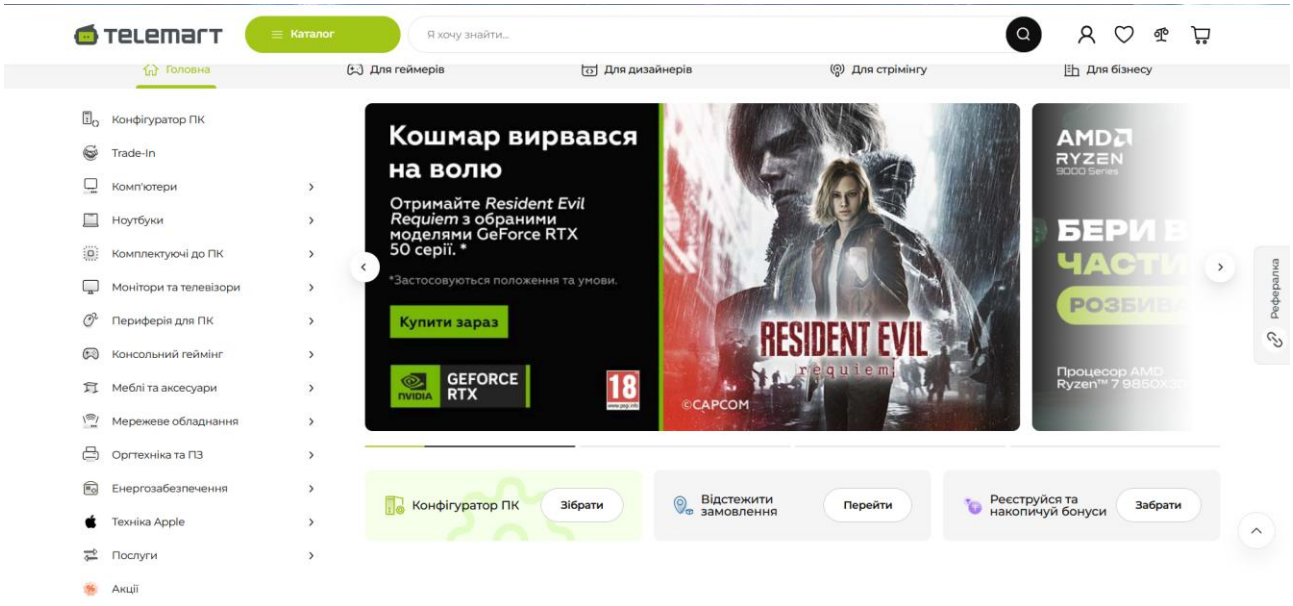


Рисунок 1.2 – Головна сторінка інтернет-магазину Telemart

Переваги:

– глибока спеціалізація на комп'ютерній техніці з детальними технічними характеристиками відеокарт;

– наявність вбудованого модуля перевірки базової сумісності обладнання при формуванні замовлення;

– повний цикл електронної комерції (від перегляду каталогу до оплати та інтеграції зі складом).

Недоліки:

– це закрита пропрієтарна система конкурента, яку неможливо придбати, скопіювати чи адаптувати для потреб магазину «V-Core»;

– вбудований конфігуратор орієнтований переважно на збірку ПК «з нуля», а не на точковий підбір окремої відеокарти під уже існуючу систему клієнта (наприклад, з урахуванням обмежень старого блоку живлення чи габаритів корпусу).

Третім аналогом є великі маркетплейси з широким асортиментом комп'ютерних комплектуючих, яскравим прикладом яких є онлайн-платформа Rozetka [4]. Вона володіє величезним каталогом товарів і є стандартом масової електронної комерції.

Головна сторінка розділу комп'ютерних комплектуючих на платформі Rozetka зображена на рисунку 1.3

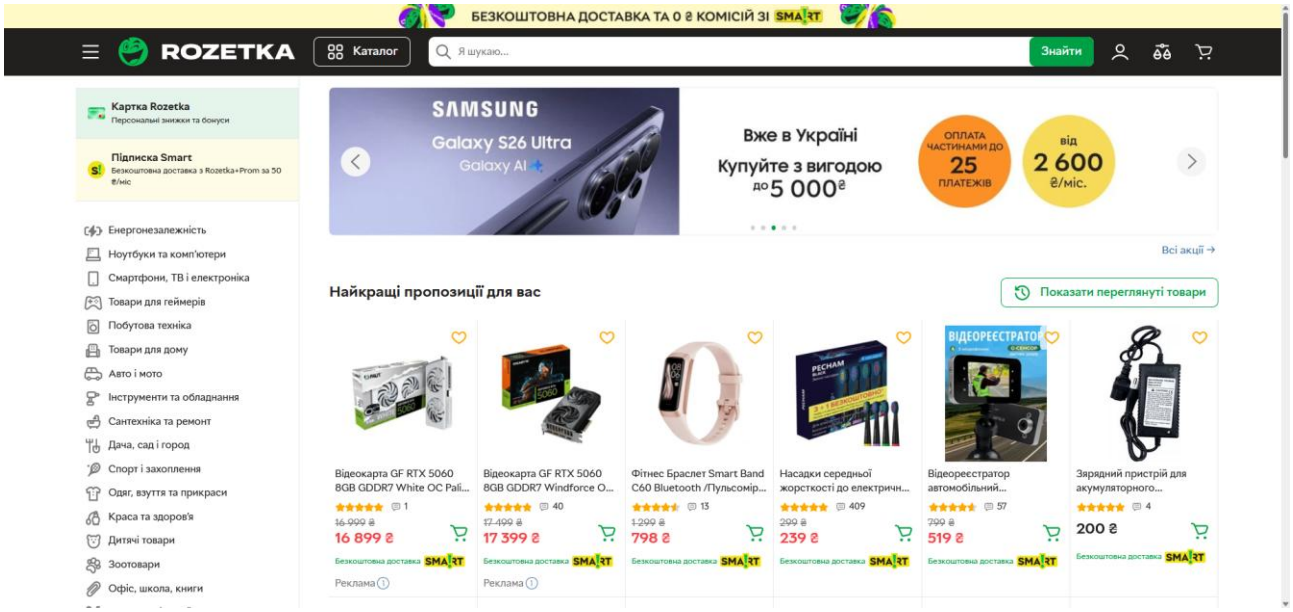


Рисунок 1.3 – Розділ комп'ютерних комплектуючих на платформі Rozetka

Переваги:

– величезна база товарів та розгалужена система фільтрів за базовими характеристиками (бренд, обсяг пам'яті, ціна);

– потужна система управління відгуками, рейтингами та особистим кабінетом покупця;

– зручний візуальний інтерфейс каталогу.

Недоліки:

– стандартна логіка фільтрації є суто лінійною і не попереджає користувача про технічну несумісність (система не підкаже, що для обраної топової відеокарти не вистачить потужності слабкого блоку живлення);

– орієнтованість на масового споживача призводить до перевантаженості інтерфейсу та відсутності вузькоспеціалізованого підходу, необхідного для продажу специфічного IT-обладнання;

– платформа не надає свого програмного ядра для розгортання незалежних магазинів із власною бізнес-логікою.

Отже, розгляд аналогів показує, що існуючі рішення або є інформаційними агрегаторами без можливості управління продажами (PCPartPicker), або закритими платформами конкурентів і масових маркетплейсів (Telemart, Rozetka). Необхідність розробки власного програмного забезпечення для автоматизації бізнес-процесів магазину «V-Core», яке повністю відповідатиме специфічним потребам цього бізнесу, є цілком актуальною.

Запропоноване рішення дозволить створити повноцінний інтернет-магазин з системою фільтрації та технічного підбору відеокарт з урахуванням сумісності комплектуючих. Це зменшить обсяг ручної роботи менеджерів, мінімізує ймовірність помилок при формуванні замовлень, скоротить час їх обробки та забезпечить централізоване зберігання даних про товари і клієнтів. Клієнти, у свою чергу, отримають цілодобовий доступ до каталогу та можливість самостійно перевіряти сумісність компонентів.

Таким чином, аналіз практичної задачі інженерії програмного забезпечення для магазину «V-Core» свідчить про доцільність розробки власного вебзастосунку, який автоматизує ключові процеси продажу комп'ютерної техніки, відкриє новий ефективний канал онлайн-збуту та сприятиме підвищенню загальної конкурентоспроможності магазину на ринку.

Створення індивідуального програмного продукту дасть змогу підвищити ефективність роботи магазину, зокрема: об'єднати каталог товарів і систему замовлень у єдиний інструмент, забезпечити точність вибору за критичними

параметрами (архітектура GPU, TDP, порти) та оптимізувати користувацький досвід покупців.

1.3 Постановка задачі на розробку програмного забезпечення для автоматизації обліку онлайн-замовлень та підбору відеокарт у магазині «V-Core»

Зважаючи на результати проведеного аналізу практичної задачі інженерії програмного забезпечення, з метою підвищення рівня автоматизації бізнес-процесів роздрібної торгівлі в магазині «V-Core» було прийняте рішення розробити власне програмне забезпечення. Його призначення полягає в оптимізації роботи менеджера та адміністратора магазину, а також у спрощенні процесу підбору відеокарт та ведення обліку продажів.

Вимоги до складу виконуваних функцій:

Додатком будуть користуватися три основні групи користувачів: адміністратор (admin) та менеджер (moderator) в адміністративній панелі, а також клієнт (customer) через публічну вітрину магазину. В адміністративній панелі менеджер може виконувати усі функції адміністратора крім:

– додавання та редагування інформації про товари в каталозі (тільки адміністратор);

– перегляд та видалення зареєстрованих менеджерів (тільки адміністратор).

Нижче всі функції, які доступні адміністратору:

– реєстрація співробітників;

Вхідні дані: логін, пароль, роль.

Результат: створення нового облікового запису в системі.

– авторизація користувачів;

Вхідні дані: логін, пароль, роль (підтягується автоматично згідно з даними вже зареєстрованого користувача).

Результат: вхід у систему з правами доступу ролі, яка була визначена під час реєстрації (admin/ moderator).

– обробка інформації про товари на складі (додавання, видалення, редагування);

Вхідні дані: артикул товару, фото, назва, технічні характеристики (архітектура GPU, обсяг пам'яті, шина, TDP, інтерфейси), кількість, ціна.

Результат: відображення, додавання, видалення або редагування товарів у каталозі (додавання/редагування доступне тільки admin).

– обробка інформації про клієнтів (додавання, видалення, редагування);

Вхідні дані: прізвище, ім'я, телефон, електронна пошта, адреса доставки.

Результат: додавання, видалення або редагування даних клієнтів.

– підбір та фільтрація відеокарт;

Вхідні дані: параметри фільтрації (бажаний обсяг пам'яті, ліміт TDP блоку живлення, виробник, діапазон цін).

Результат: відображення повного списку відеокарт, відображення списку відеокарт, що відповідають заданим критеріям сумісності та вимогам клієнта.

– створення замовлення на покупку (додавання усієї інформації щодо продажу, пошук товару та клієнта, внесення нового клієнта у разі його відсутності);

Вхідні дані: номер замовлення, ПІБ клієнта, дата замовлення, статус оплати, менеджер, що оформлював покупку, спосіб доставки, перелік товарів, загальна сума.

Результат: створення нового запису продажу зі списанням товару зі складу.

– редагування та видалення замовлення;

Вхідні дані: номер замовлення, ПІБ клієнта, дата замовлення, статус оплати, перелік товарів, загальна сума.

Результат: редагування та видалення замовлення з відповідним поверненням товару на баланс.

– оновлення статусу замовлення на «виконано» або «відправлено»;

Вхідні дані: новий статус.

Результат: оновлення статусу замовлення в базі даних для подальшого відстеження.

– відображення всього списку товарів, доданих у конкретне замовлення;

Вхідні дані: id замовлення.

Результат: відображення детальної специфікації товарів у кошику клієнта.

– перегляд та видалення зареєстрованих менеджерів (доступне тільки admin);

Вхідні дані для видалення: id moderator.

Результат для видалення: відображення списку менеджерів без обраного (видаленого) співробітника.

– вихід з облікового запису;

Вхідні дані: немає.

Результат: завершення сесії адміністратора/менеджера.

Окрім функцій для співробітників, у системі передбачено публічну частину для клієнтів (покупців). Клієнтам доступні наступні функції:

– реєстрація та авторизація клієнта;

Вхідні дані: електронна пошта, пароль.

Результат: створення особистого облікового запису клієнта та вхід у систему.

– перегляд каталогу та підбір відеокарт;

Вхідні дані: параметри фільтрації (бажаний обсяг пам'яті, ліміт TDP, виробник, ціновий діапазон).

Результат: відображення списку товарів, що відповідають запиту.

– управління кошиком покупок;

Вхідні дані: ідентифікатор товару, кількість.

Результат: додавання товару до кошика, зміна кількості, видалення товару, автоматичний перерахунок загальної вартості.

– оформлення замовлення;

Вхідні дані: контактні дані, адреса доставки, спосіб оплати, перелік товарів у кошику.

Результат: створення нового замовлення в системі (зі статусом «нове» або «очікує обробки») та очищення кошика.

– перегляд історії замовлень (в особистому кабінеті);

Вхідні дані: ідентифікатор авторизованого клієнта.

Результат: відображення списку поточних та попередніх замовлень клієнта з їхніми статусами.

Вимоги до організації вхідних та вихідних даних:

Усі постійні дані системи (каталог відеокарт, облікові записи користувачів, історія замовлень, відгуки) повинні зберігатися у реляційній базі даних SQLite.

Вхідні дані:

введення даних здійснюється користувачами (клієнтами, менеджерами, адміністратором) через графічний вебінтерфейс за допомогою стандартних пристроїв (клавіатура, миша, сенсорні екрани мобільних пристроїв). Дані можуть:

– вводитися вручну через текстові поля (наприклад, ПІБ, адреса доставки, технічні характеристики відеокарт, текст відгуку);

– обиратися з наявних випадаючих списків, чекбоксів або радіокнопок (наприклад, вибір бренду, типу пам'яті, статусу замовлення, способу оплати);

– задаватися за допомогою інтерактивних елементів (наприклад, повзунки для вибору цінового діапазону у фільтрах підбору відеокарт).

– автоматично завантажуватися з бази даних системи (наприклад, збережені контактні дані авторизованого клієнта, історія попередніх замовлень, актуальні залишки товарів на складі).

Вихідні дані:

виведення даних відбувається на екранах пристроїв користувачів у режимі реального часу після обробки запитів сервером. Формати вихідних даних включають:

– динамічно згенеровані вебсторінки з каталогом товарів та детальними характеристиками відеокарт;

– табличні представлення даних (для зручного перегляду історії замовлень, списку клієнтів або інвентаризації складу в панелі керування).

Вимоги до складу та параметрів технічних засобів:

Оскільки система має клієнт-серверну архітектуру, технічні вимоги поділяються на вимоги до робочого місця та вимоги до сервера.

Для повноцінної роботи з панеллю керування (робоче місце менеджера/адміністратора) необхідна система наступної мінімальної конфігурації:

– процесор: двоядерний із тактовою частотою не менше 2,0 ГГц;

– оперативна пам'ять: від 4 ГБ;

– системний диск: 100 ГБ вільного дискового простору (для операційної системи, локального кешу та збереження вивантажених звітів);

– монітор: роздільна здатність не менше 1920x1080 пікселів для комфортної роботи з об'ємними таблицями замовлень та каталогом відеокарт;

– необхідно стабільне мережеве з'єднання з виходом в інтернет.

Для клієнтів (покупців вітрини) мінімальні вимоги зводяться до наявності будь-якого сучасного пристрою (ПК, ноутбук, планшет або смартфон) з підключенням до мережі Інтернет.

Для розгортання серверної частини (мінімальні параметри хостингу або VPS-сервера):

– процесор: від 1 до 2 віртуальних ядер (vCPU);

– оперативна пам'ять: мінімум 2 ГБ (рекомендовано 4 ГБ для стабільної роботи бекенду та бази даних);

– дисковий простір: від 20 ГБ SSD для зберігання БД та медіафайлів (зображень товарів).

Вимоги до інформаційної та програмної сумісності:

– сумісність з операційними системами на клієнтських робочих місцях: операційна система не нижче Windows 10, macOS або сучасні дистрибутиви Linux;

– сумісність з базами даних: наявність системи управління базами даних SQLite (версії 3.x або вище), що використовується для зберігання даних каталогу, користувачів та замовлень;

– сумісність з веббраузерами: оскільки програма має вебінтерфейс, вона повинна бути повністю сумісною та коректно відображатися у сучасних веббраузерах, таких як Google Chrome, Mozilla Firefox, Opera, Safari та Microsoft Edge.

Для розгортання серверної частини вебзастосунку використовується хостинг (або VPS-сервер) із встановленою панеллю керування FastPanel або аналогічним програмним забезпеченням, що забезпечує зручне адміністрування серверних ресурсів [5]. Основними вимогами до серверного середовища виконання є:

– наявність стабільного інтернет-з’єднання, оскільки вся взаємодія із системою (обробка замовлень, оновлення каталогу) відбувається онлайн;

– сервер із параметрами, достатніми для коректної роботи вебдодатку (рекомендовано принаймні 1-2 віртуальних ядра процесора (vCPU), 2-4 ГБ оперативної пам’яті та від 20 ГБ вільного дискового простору SSD);

– операційна система на сервері, сумісна з обраними технологіями (наприклад, Linux-дистрибутиви Ubuntu або Debian);

– наявність підтримуваної версії вебсервера (Nginx або Apache), що забезпечує обробку HTTP-запитів, швидку та безпечну роздачу скомпільованих статичних файлів клієнтської частини (React-застосунку, зібраного за допомогою Vite) та маршрутизацію (Reverse Proxy) API-запитів;

– підтримка необхідного середовища виконання для бекенд-частини (наприклад, середовище Node.js для забезпечення роботи REST API, обробки бізнес-логіки та взаємодії з базою даних).

Крім того, для повноцінної роботи з вебзастосунком на стороні користувача (клієнта або менеджера) потрібно мати пристрій (ПК, ноутбук або смартфон) із сучасним веббраузером, що має актуальні оновлення та доступ до мережі Інтернет.

Детальна постановка задачі представлена у технічному завданні, яке наведено у додатку А.

2 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ДЛЯ МАГАЗИНУ «V-CORE»

2.1 Аналіз вимог до програмного забезпечення

Аналіз вимог є фундаментальним етапом життєвого циклу розробки програмного забезпечення, що дозволяє визначити функціональні межі системи та очікування зацікавлених сторін. Для формалізації функціональних вимог до вебзастосунку магазину «V-Core» було застосовано метод моделювання варіантів використання (Use Case modeling). Цей підхід дозволяє описати поведінку системи з точки зору її користувачів (зокрема клієнтів, менеджерів та адміністраторів), ідентифікуючи ключових акторів та їхні цілі у процесах підбору комп'ютерних комплектуючих, оформлення замовлень та управління асортиментом.

2.1.1 Розробка моделі варіантів використання

Першим кроком у проєктуванні системи є ідентифікація акторів – зовнішніх сутностей, що взаємодіють із вебзастосунком. Відповідно до технічного завдання, для вебзастосунку магазину «V-Core» було виявлено три основні групи користувачів, ролі яких описано в таблиці 2.1.

Таблиця 2.1 – Опис виявлених акторів системи

Актор	**Опис ролі**
1	**2**
Клієнт (Customer)	Користувач системи, який здійснює підбір, порівняння та перегляд каталогу відеокарт, використовує фільтрацію, керує віртуальним кошиком, публікує коментарі та оформлює замовлення.

Кінець таблиці 2.1

1	2
Менеджер (Moderator)	Співробітник магазину, який працює в адміністративній панелі: здійснює обробку замовлень клієнтів (оновлює їх статуси) та виконує модерацію користувацьких коментарів до товарів.
Адміністратор (Admin)	Головний користувач із повним доступом. Успадковує функції менеджера та додатково здійснює керування каталогом (CRUD), налаштування параметрів сумісності, управління персоналом та формування звітності.

На основі визначених ролей акторів було сформовано детальний перелік основних прецедентів (варіантів використання) системи. Це дозволяє чітко окреслити функціональні межі вебзастосунку перед побудовою графічної моделі. Опис прецедентів наведено в таблиці 2.2.

Таблиця 2.2 – Виявлені прецеденти системи

№	Назва варіанту використання	Короткий опис
1	**2**	**3**
1	Підбір відеокарт	Використання алгоритму інтелектуального підбору відеокарт на основі заданих параметрів, вимог до продуктивності або конфігурації ПК користувача.
2	Перегляд каталогу відеокарт	Ознайомлення відвідувачів із доступним асортиментом, технічними характеристиками та актуальними цінами товарів магазину.
3	Додавання товару до кошику	Вибір конкретної моделі відеокарти з каталогу та додавання товару до віртуального кошика для подальшого оформлення замовлення.

Продовження таблиці 2.2

1	2	3
4	Фільтрація та сортування відеокарт	Динамічний відбір товарів за параметрами (бренд, обсяг пам’яті, ціна) для зручного пошуку необхідної моделі.
5	Публікація коментарів	Написання відгуків та виставлення оцінок товарам. Функція розширює можливості перегляду каталогу та доступна авторизованим користувачам.
6	Авторизація	Вхід у систему згідно з правами доступу (клієнт, менеджер, адміністратор) для отримання доступу до розширених функцій застосунку.
7	Реєстрація	Створення нового облікового запису в системі для збереження історії замовлень та взаємодії з контентом. Розширює процес авторизації.
8	Перегляд кошику	Відображення переліку доданих товарів, їх кількості, актуальної ціни та загальної вартості майбутнього замовлення.
9	Редагування кошику	Зміна кількості обраних товарів або видалення окремих позицій відеокарт з віртуального кошика.
10	Оформлення замовлень	Процес підтвердження контактних даних, вибору способу доставки та завершення процедури онлайн-покупки на основі вмісту кошика.
11	Порівняння відеокарт	Зіставлення технічних характеристик кількох обраних моделей відеокарт для аналізу та вибору оптимального варіанту.
12	Обробка замовлень	(Менеджер, Адміністратор) Перегляд нових замовлень, зміна їх статусів (наприклад, «Відправлено», «Виконано») та загальне ведення продажів.

Кінець таблиці 2.2

1	2	3
13	Модерація коментарів	(Менеджер, Адміністратор) Керування користувацьким контентом: перегляд, схвалення або видалення відгуків до товарів.
14	Керування каталогом (CRUD)	(Адміністратор) Повний цикл управління товарами: додавання нових карток відеокарт, редагування описів, цін та видалення.
15	Управління персоналом	(Адміністратор) Адміністрування облікових записів персоналу магазину та налаштування рівнів доступу (призначення менеджерів).
16	Формування статистики та звітності	(Адміністратор) Формування аналітичних даних щодо продажів, популярності окремих моделей відеокарт та загальних показників.
17	Налаштування параметрів сумісності	(Адміністратор) Керування правилами алгоритму інтелектуального підбору (встановлення лімітів TDP, габаритів та вимог до БЖ).

Для візуалізації взаємодії між ідентифікованими акторами (таблиця 2.1) та прецедентами (таблиця 2.2) була розроблена діаграма варіантів використання (рисунок 2.1). Вона демонструє логічні межі вебзастосунку «V-Core» та зв'язки між функціональними блоками.

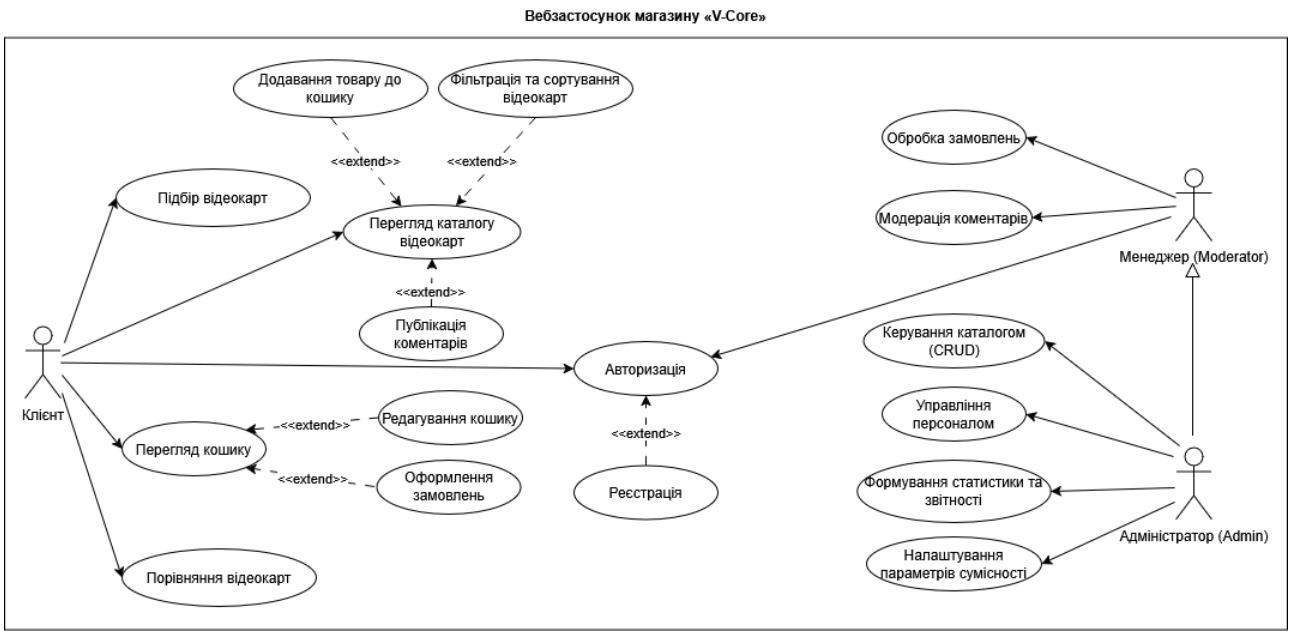


Рисунок 2.1 – Діаграма варіантів використання вебзастосунку магазину «V-Core»

Таким чином, представлена модель варіантів використання формує повний набір функціональних вимог до публічної частини та адміністративної панелі системи, що є основою для подальшого проєктування бази даних та архітектури застосунку.

2.1.2 Розробка специфікацій варіантів використання

Після ідентифікації акторів та основних прецедентів необхідно деталізувати кожен варіант використання. Специфікація прецеденту дозволяє формалізувати вимоги, описуючи послідовність дій, передумови та постумови для успішного виконання функції. Детальні специфікації всіх ключових прецедентів системи, починаючи з варіанту використання «Підбір відеокарт», наведено у таблицях 2.3 – 2.9. Повний комплекс специфікацій для всіх варіантів використання, які деталізують функціональні межі публічної вітрини та адміністративної панелі вебзастосунку «V-Core», винесено до відповідного розділу Технічного завдання (Додаток А, підрозділ 3.1).»

Таблиця 2.3 – Специфікація варіанту використання «Підбір відеокарт»

Характеристика	Опис
Короткий опис	Автоматична перевірка сумісності комплектуючих із системою користувача (блок живлення (БЖ) та корпус) або інтелектуальний підбір за заданими параметрами
Дійові особи	Клієнт
Передумови	Наявність товарів у базі даних із заповненими полями TDP та довжини
Основний потік подій	1. Користувач переходить у розділ «Підбір». 2. Вводить потужність свого БЖ (у ватах) та макс. довжину (у мм). 3. Натискає «Підібрати». 4. Алгоритм відсіює моделі, чий TDP + запас перевищує БЖ, або які довші за вказаний ліміт. 5. Відображається список 100% сумісних карток.
Альтернативний потік подій	2а. Введено некоректні дані (наприклад, від'ємні значення) – клієнтська валідація блокує запит.
Постумови	Користувач отримує технічно обґрунтовані пропозиції

Детальне ознайомлення з асортиментом магазину формалізовано у наступній специфікації (таблиця 2.4).

Таблиця 2.4 – Специфікація варіанту використання «Перегляд каталогу відеокарт»

Характеристика	Опис
1	**2**
Короткий опис	Користувач переглядає загальний асортимент магазину з фотографіями, цінами та основними характеристиками

Кінець таблиці 2.4

1	2
Дійові особи	Клієнт
Передумови	База даних доступна, каталог містить хоча б один товар
Основний потік подій	1. Користувач відкриває головну сторінку або розділ «Каталог». 2. Система відправляє запит до БД на отримання списку товарів. 3. Система відображає сітку карток товарів (пагінація). 4. Користувач може натиснути на товар для перегляду деталей.
Альтернативний потік подій	2а. Помилка з'єднання з БД – система виводить повідомлення: «Не вдалося завантажити товари».
Постумови	Користувач ознайомлений з асортиментом

Процес відбору товарів для майбутньої покупки регламентується таблицею 2.5.

Таблиця 2.5 – Специфікація варіанту використання «Додавання товару до кошику»

Характеристика	Опис
1	**2**
Короткий опис	Вибір конкретної моделі відеокарти та додавання її до віртуального кошика для подальшої покупки
Дійові особи	Клієнт
Передумови	Користувач знаходиться в каталозі або на сторінці конкретного товару

Кінець таблиці 2.5

1	2
Основний потік подій	1. Користувач натискає кнопку «Додати до кошика» біля обраного товару. 2. Система фіксує додавання товару в поточну сесію. 3. Оновлюється індикатор кількості товарів на іконці кошика. 4. Виводиться спливаюче повідомлення про успішне додавання.
Альтернативний потік подій	1а. Товар закінчився на складі – кнопка неактивна або виводиться відповідне повідомлення.
Постумови	Обраний товар знаходиться у віртуальному кошику користувача

Логіку динамічного пошуку та відбору товарів описано у специфікації прецеденту «Фільтрація та сортування відеокарт» (таблиця 2.6).

Таблиця 2.6 – Специфікація варіанту використання «Фільтрація та сортування відеокарт»

Характеристика	**Опис**
1	**2**
Короткий опис	Динамічний відбір товарів за специфічними технічними параметрами (бренд, пам'ять) та ціною
Дійові особи	Клієнт
Передумови	Користувач перебуває на сторінці каталогу

Кінець таблиці 2.6

1	2
Основний потік подій	1. Користувач обирає критерії на панелі фільтрів (наприклад, NVIDIA, 12 ГБ). 2. Обирає тип сортування (наприклад, «Від дешевих до дорогих»). 3. Система формує параметризований запит. 4. Перелік товарів миттєво оновлюється без перезавантаження сторінки.
Альтернативний потік подій	4а. За обраними фільтрами товарів не знайдено – виводиться повідомлення: «Товарів не знайдено. Спробуйте змінити критерії».
Постумови	На екрані відображаються лише релевантні запиту товари

Можливість взаємодії користувачів із платформою через зворотний зв'язок деталізовано в таблиці 2.7.

Таблиця 2.7 – Специфікація варіанту використання «Публікація коментарів»

Характеристика	**Опис**
1	**2**
Короткий опис	Написання текстового відгуку та виставлення рейтингу конкретному товару
Дійові особи	Клієнт
Передумови	Клієнт авторизований

Кінець таблиці 2.7

1	2
Основний потік подій	1. Клієнт відкриває картку товару. 2. Пише текст відгуку та обирає оцінку (від 1 до 5 зірок). 3. Натискає «Опублікувати». 4. Коментар зберігається в БД і з'являється на сторінці товару. 5. Середній рейтинг відеокарти перераховується.
Альтернативний потік подій	2а. Спроба відправити порожній коментар – кнопка блокується системою валідації.
Постумови	Відгук додано до загального списку коментарів товару

Для отримання доступу до розширених функцій системи користувач повинен пройти процес ідентифікації, який регламентується таблицею 2.8.

Таблиця 2.8 – Специфікація варіанту використання «Авторизація»

Характеристика	**Опис**
1	**2**
Короткий опис	Вхід у систему для отримання доступу до персоналізованих або адміністративних функцій
Дійові особи	Клієнт, Менеджер, Адміністратор
Передумови	Обліковий запис попередньо створено у базі даних

Кінець таблиці 2.8

1	2
Основний потік подій	1. Користувач відкриває форму авторизації. 2. Вводить логін (email) та пароль. 3. Система валідує хеш пароля. 4. Система визначає роль та генерує JWT-токен доступу. 5. Відбувається перенаправлення до особистого кабінету або адмін-панелі.
Альтернативний потік подій	3а. Невірний пароль або логін – виводиться повідомлення про помилку.
Постумови	Сесія розпочата, права доступу застосовано

Процес створення нового облікового запису для неавторизованих клієнтів описано в таблиці 2.9.

Таблиця 2.9 – Специфікація варіанту використання «Реєстрація»

Характеристика	Опис
Короткий опис	Створення нового клієнтського облікового запису
Дійові особи	Клієнт
Передумови	Користувач не авторизований у системі
Основний потік подій	1. Користувач відкриває сторінку реєстрації. 2. Заповнює форму (ПІБ, Email, Пароль). 3. Система перевіряє унікальність Email. 4. Пароль хешується, дані зберігаються у БД. 5. Користувач автоматично авторизується.
Альтернативний потік подій	3а. Email вже існує в БД – система пропонує авторизуватися або відновити пароль.
Постумови	Створено новий профіль клієнта

Таким чином, деталізовані специфікації всіх ключових варіантів використання утворюють повний набір функціональних вимог, що є фундаментальною основою для подальшого етапу проєктування архітектури та розробки бази даних вебзастосунку «V-Core».

2.2 Розробка архітектури програмного забезпечення вебзастосунку «V-Core»

Розробка архітектури є ключовим етапом проєктування, на якому визначається загальна структура системи, її основні компоненти та взаємозв'язки між ними. Правильно обрана архітектура забезпечує надійність, масштабованість, гнучкість та легкість супроводу програмного продукту. Для вебсистеми магазину «V-Core» було обрано сучасну клієнт-серверну архітектуру з чітким розділенням на клієнтську частину (Single Page Application) та серверну частину (RESTful API), що дозволяє незалежно розробляти та масштабувати представлення даних і бізнес-логіку.

На рисунку 2.2 представлена діаграма компонентів, яка візуалізує логічну структуру вебзастосунку. Вона ілюструє взаємодію двох основних рівнів системи. Клієнтська частина містить UI-компоненти (реалізовані за допомогою React та Tailwind CSS), модулі управління станом та API-клієнт для зв'язку з сервером. Серверна частина (Node.js) включає маршрутизатори та контролери (API Routers / Controllers) для обробки вхідних HTTP-запитів, шар бізнес-логіки (Services) для виконання таких операцій, як перевірка технічної сумісності відеокарт та обробка замовлень, а також шар доступу до даних (Data Access Layer / ORM) для взаємодії з реляційною базою даних SQLite та виконання CRUD-операцій.

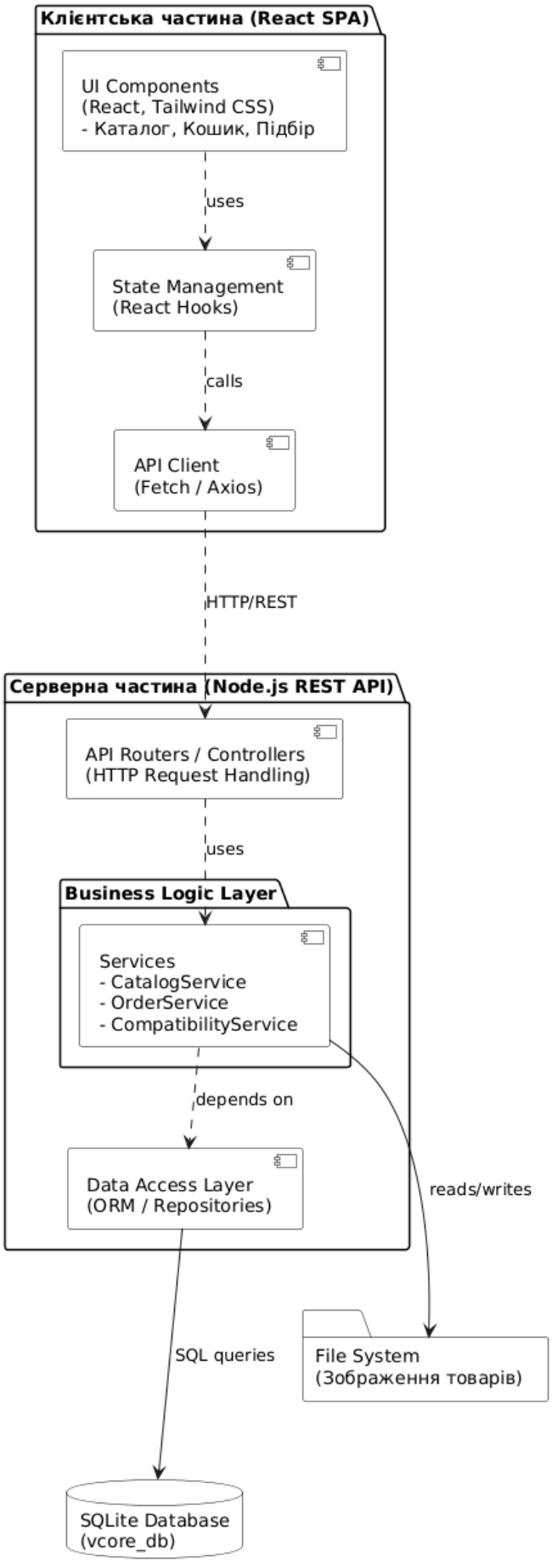


Рисунок 2.2 – Діаграма компонентів вебзастосунку магазину «V-Core»

Для розуміння фізичного розміщення компонентів системи була розроблена діаграма розгортання (рисунок 2.3). Вона показує, як програмні артефакти розподілені по вузлах інфраструктури. Клієнтський вузол містить веббраузер користувача, де виконується клієнтська частина застосунку (Single Page Application, створений за допомогою React та TypeScript і скомпільований у базові HTML, CSS та JavaScript). Серверний вузол (VPS/Хостинг) включає вебсервер для швидкої роздачі статичних файлів фронтенду та основний бекенд-додаток (REST API на базі середовища Node.js), а вузол бази даних містить реляційну СУБД SQLite. Також на сервері виділено файлове сховище для збереження медіаданих (зображень відеокарт).

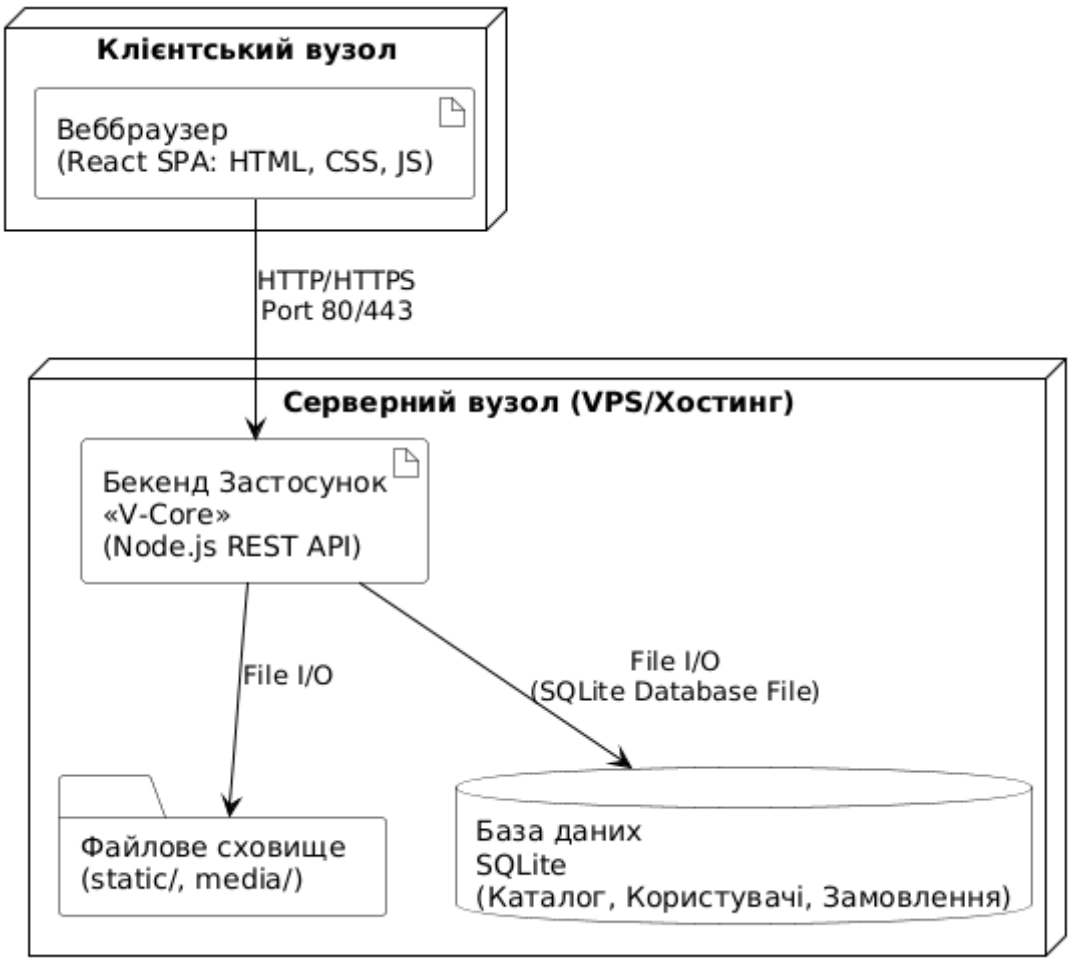


Рисунок 2.3 – Діаграма розгортання вебзастосунку магазину «V-Core»

Таким чином, обрана архітектура розгортання забезпечує гнучкість системи, дозволяючи масштабувати кожен рівень (клієнтський, серверний, базу даних) незалежно один від одного. Це підкреслює оптимізований характер застосунку, розрахований на ефективне управління каталогом товарів, швидку обробку онлайн-замовлень та надійне зберігання комерційної інформації.

2.3 Проєкт програмного забезпечення вебзастосунку «V-Core»

На основі розробленої архітектури було створено проєкт програмного забезпечення, що включає ескізний, технічний та робочий етапи.

2.3.1 Ескізний проєкт

Ескізний проєкт фокусується на візуалізації поведінки системи та розробці прототипів користувацького інтерфейсу.

Для деталізації ключового сценарію взаємодії клієнта з системою – підбору та фільтрації відеокарт – була створена діаграма діяльності (рисунок 2.4). Вона покроково ілюструє логіку процесу: від відображення сторінки каталогу та отримання параметрів від користувача (зокрема потужності наявного блоку живлення), до валідації введених даних, відправки запиту до серверного API, виконання SQL-запиту до бази даних, перевірки технічної сумісності та відображення результатів підбору у клієнтському інтерфейсі.

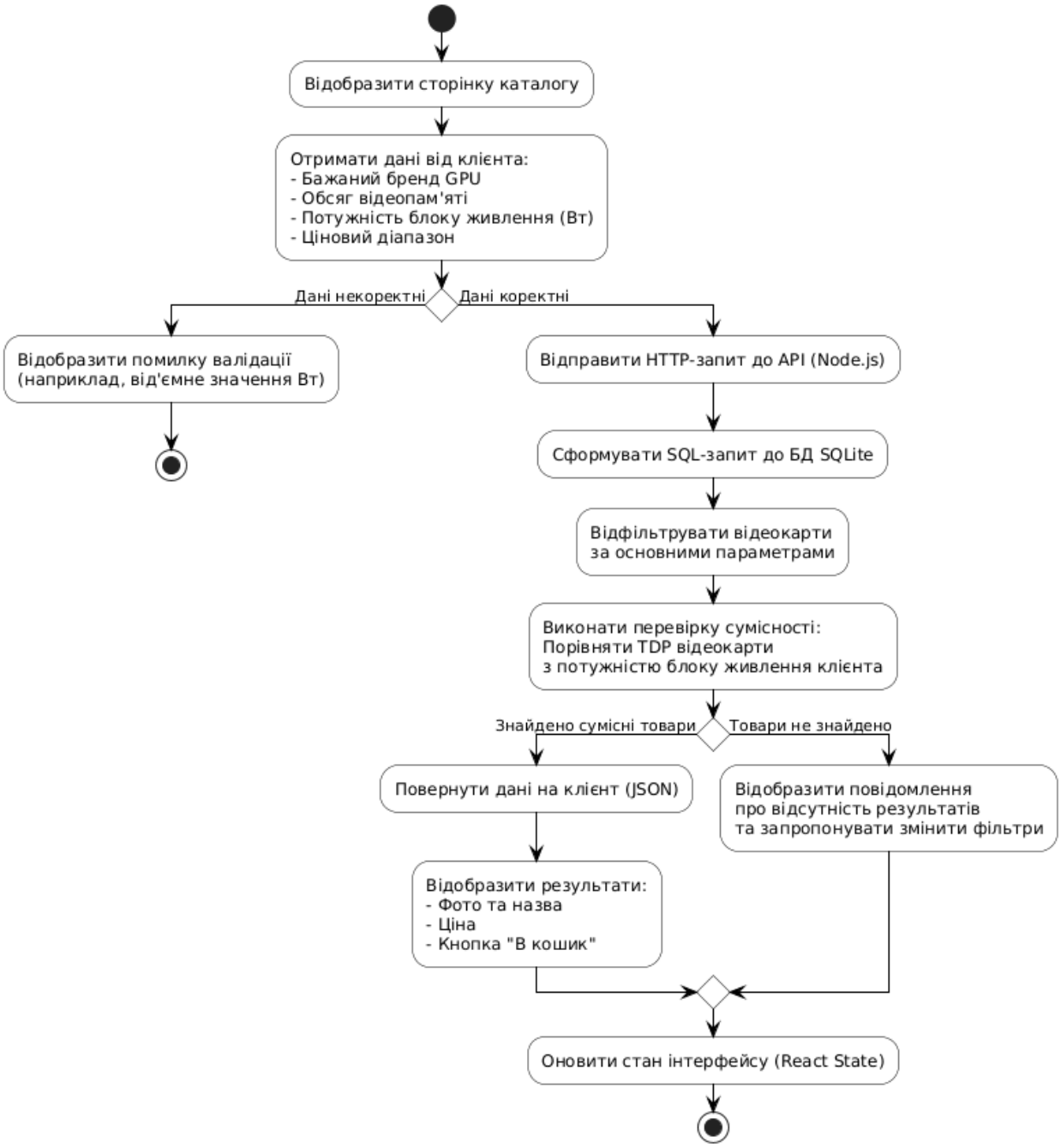


Рисунок 2.4 – Діаграма діяльності варіанту використання «Підбір та фільтрація відеокарт»

Для візуалізації користувацького досвіду та узгодження вимог до інтерфейсу було розроблено прототипи основних екранів вебзастосунку «V-Core».

На рисунку 2.5 зображено головну сторінку сайту. Вона виконує роль лендингу, представляючи логотип, основне призначення магазину та ключові

переваги системи: підбір за технічними параметрами, актуальність каталогу, швидке оформлення замовлень.

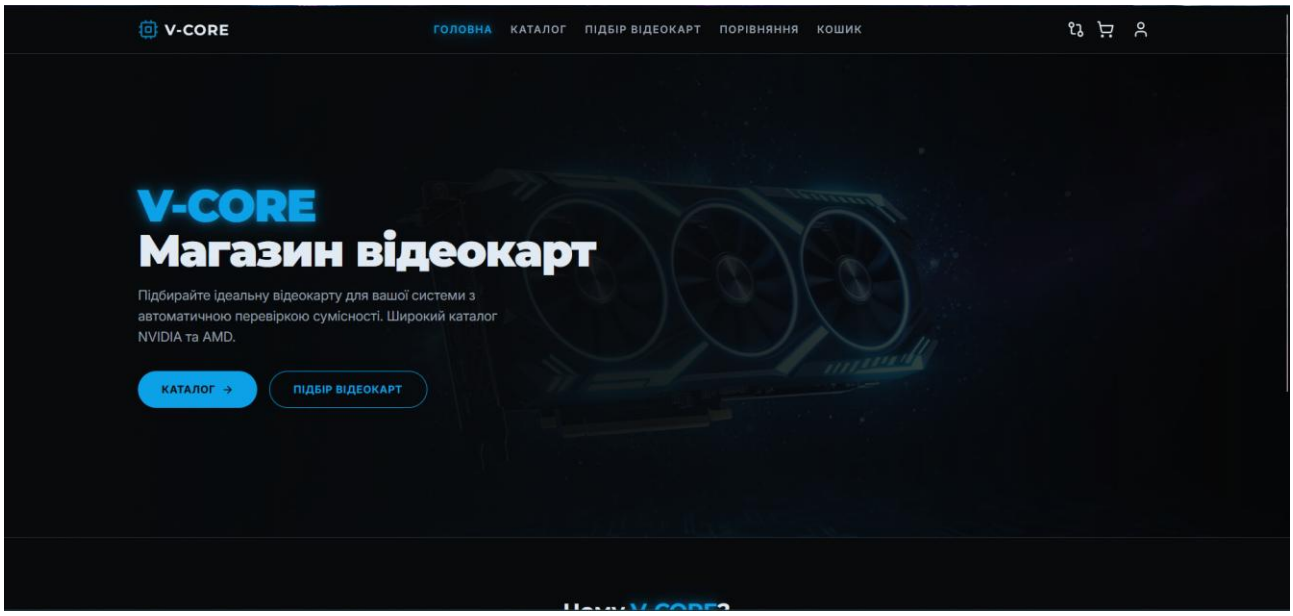


Рисунок 2.5 – Головна сторінка вебзастосунку «V-Core»

На рисунку 2.6 представлено інтерфейс загального каталогу відеокарт. Екран містить розгалужену панель фільтрації (за виробником, архітектурою, обсягом пам’яті, TDP та ціною) та функціонал сортування. Кожна картка товару відображає основні технічні характеристики, ціну та статус наявності на складі.

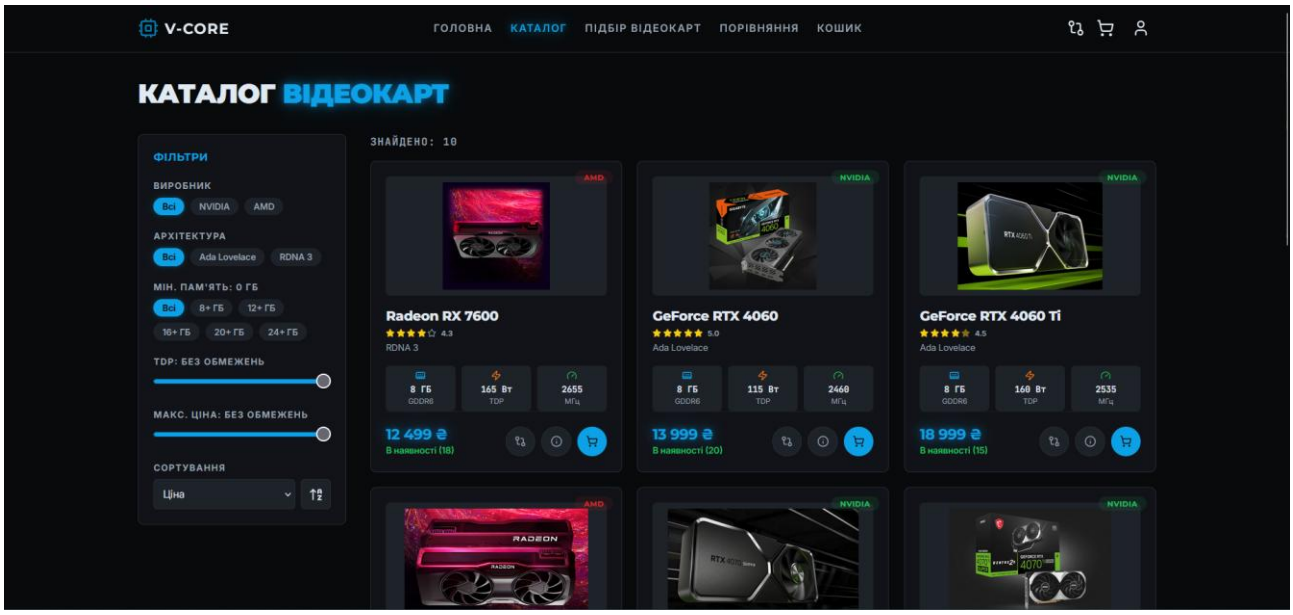


Рисунок 2.6 – Інтерфейс каталогу відеокарт

Для реалізації специфічної функціональності системи було розроблено інтерфейс підбору відеокарт (рисунок 2.7). На відміну від звичайного каталогу, цей екран фокусується на параметрах системи користувача: потужності блоку живлення (діапазон до 1200 Вт) та максимальній довжині GPU, що дозволяє автоматично відсіяти несумісні моделі.

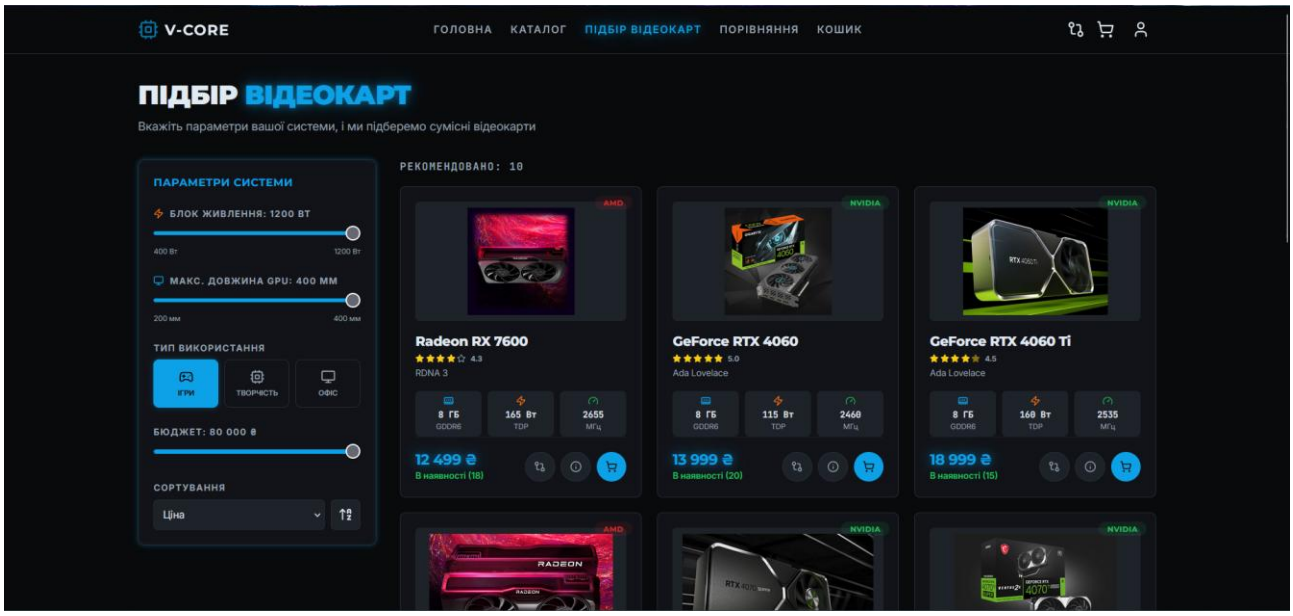


Рисунок 2.7 – Інтерфейс модуля підбору відеокарт за параметрами системи

На рисунку 2.8 продемонстровано функціонал порівняння обраних моделей. Користувач має змогу зіставити відеокарти за детальними характеристиками: архітектурою, типом пам’яті, частотами, фізичними габаритами та інтерфейсами підключення (PCIe 4.0), що полегшує прийняття рішення про купівлю.

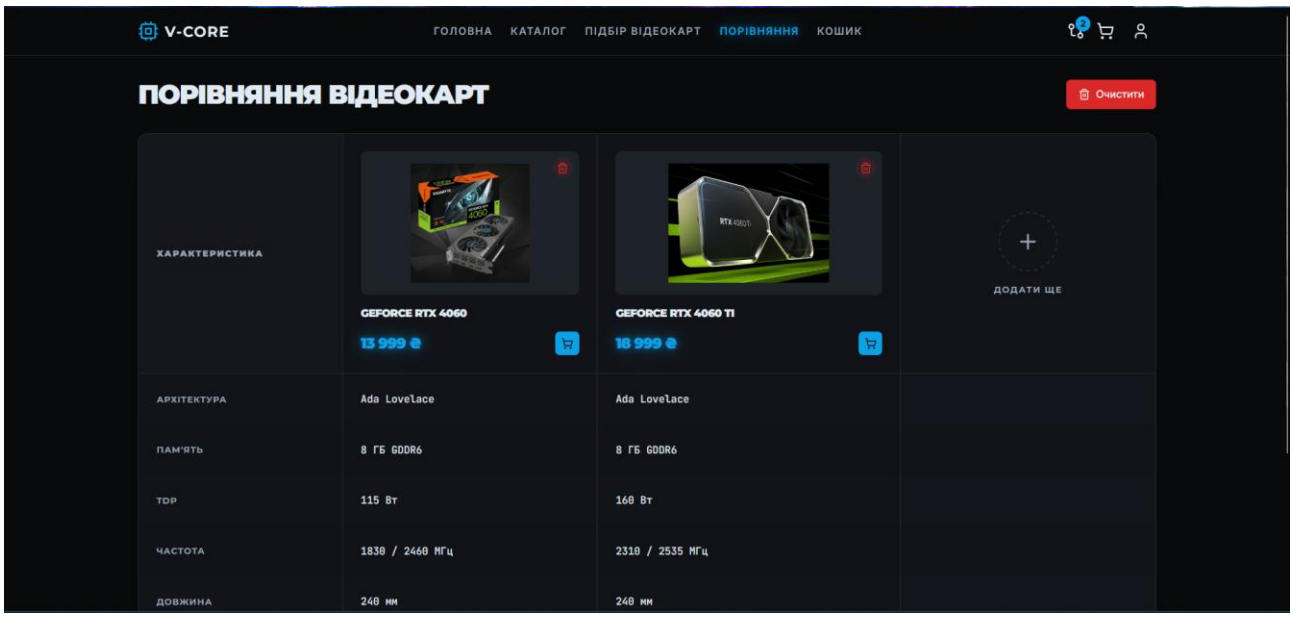


Рисунок 2.8 – Інтерфейс порівняння характеристик відеокарт

На рисунку 2.9 зображено інтерфейс кошика покупок. Він дозволяє переглядати перелік обраних товарів, їхні індивідуальні характеристики та автоматично розраховану загальну вартість перед переходом до етапу оплати.

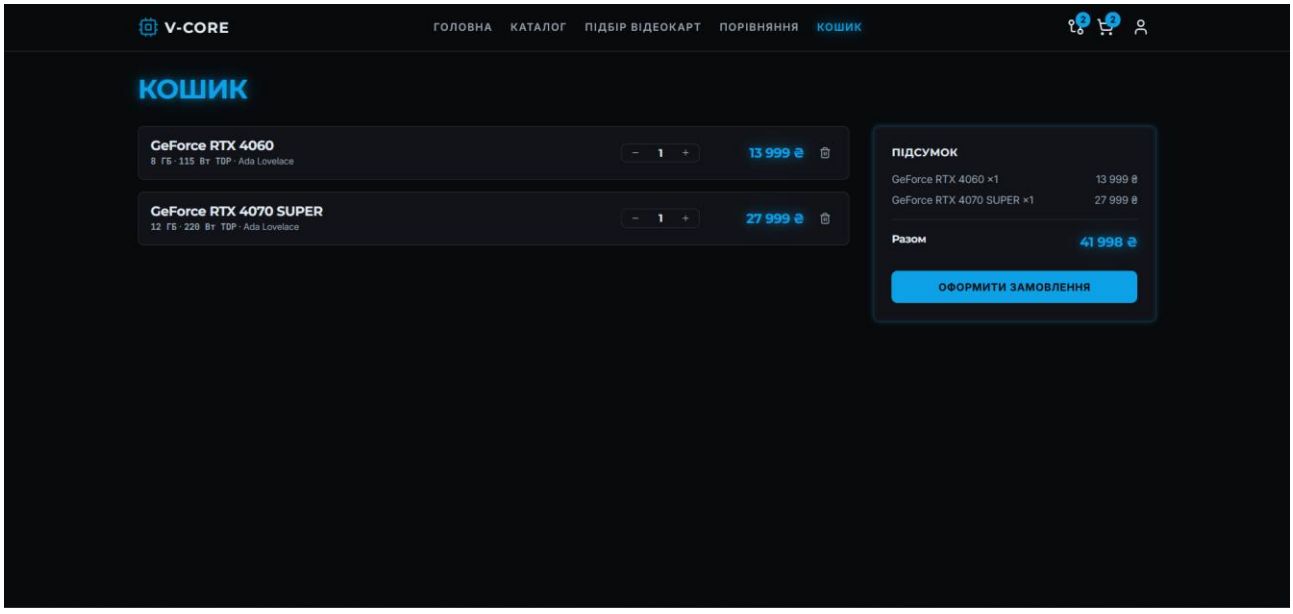


Рисунок 2.9 – Інтерфейс кошика покупок

Для забезпечення безпеки та персоналізації розроблено лаконічну форму авторизації (рисунок 2.10), яка вимагає введення імені користувача та пароля для доступу до особистого кабінету або адміністративної панелі.

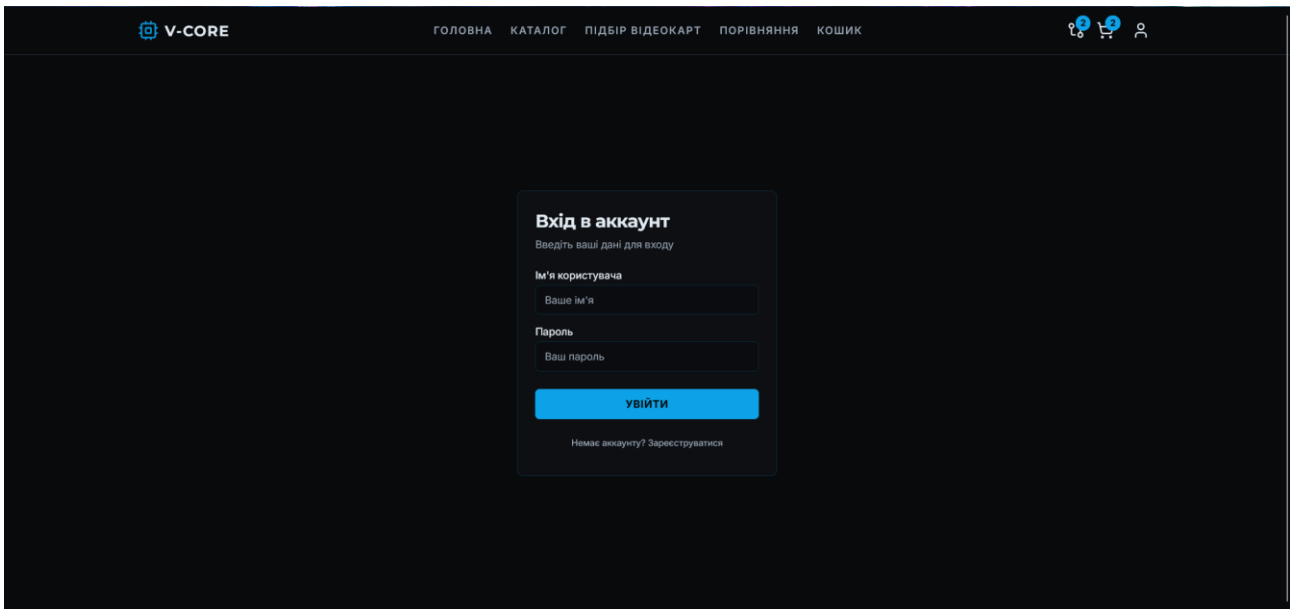


Рисунок 2.10 – Інтерфейс форми входу до системи

Адміністративна частина системи представлена двома основними модулями. На рисунку 2.11 зображено панель керування контентом, де адміністратор може виконувати CRUD-операції над каталогом: додавати нові моделі, редагувати ціни та актуалізувати статус наявності товарів.

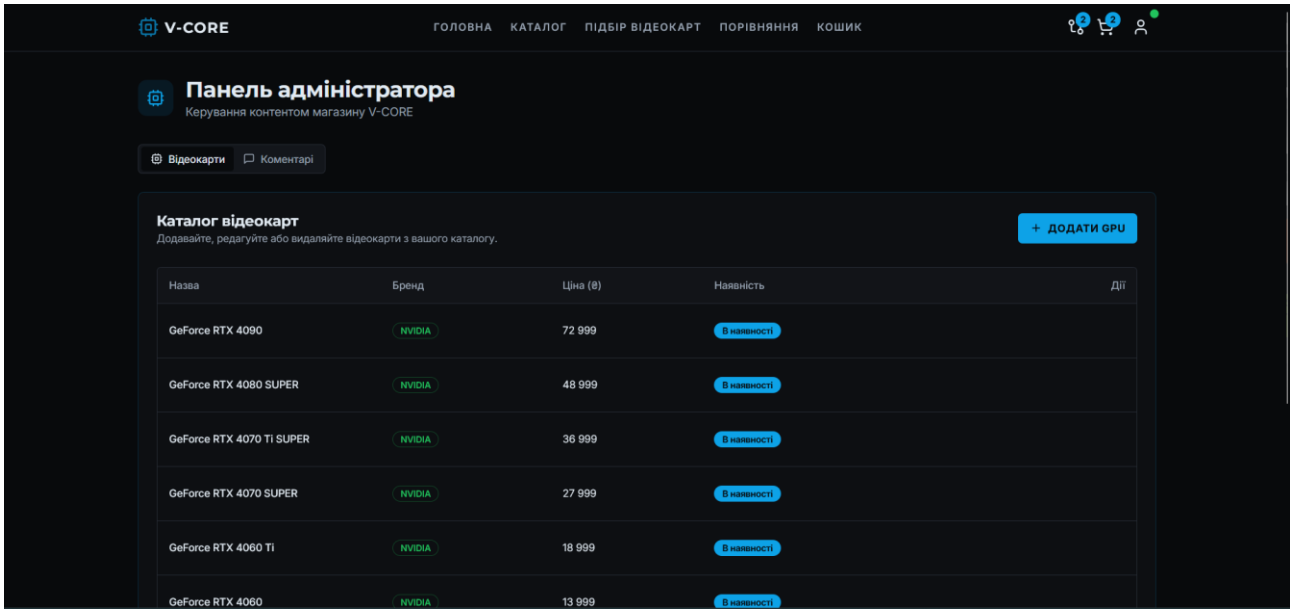


Рисунок 2.11 – Адміністративна панель: керування каталогом товарів

На рисунку 2.12 показано модуль модерації коментарів. Цей функціонал дозволяє адміністратору переглядати відгуки користувачів, їхні оцінки та

текстові повідомлення, забезпечуючи зворотний зв’язок та контроль якості контенту на платформі.

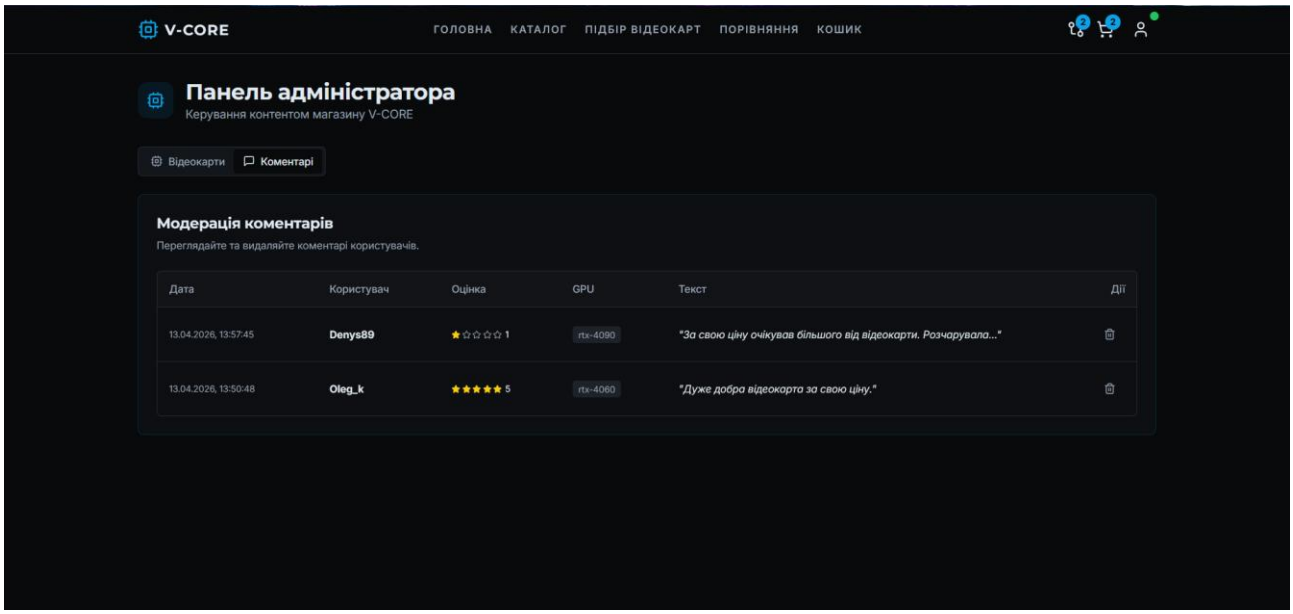


Рисунок 2.12 – Адміністративна панель: модерація відгуків користувачів

Ці прототипи інтерфейсу (рис. 2.5-2.12) демонструють інтуїтивно зрозуміле керування процесом вибору та купівлі комплектуючих, забезпечуючи зручність взаємодії клієнта з системою та ефективність інструментів адміністрування. Вони підтверджують, що функціональні вимоги до інтерфейсу, зокрема зручність перевірки апаратної сумісності та швидкість обробки замовлень, повністю відображені в ескізному проєкті.

Також під час ескізного проєктування системи була розроблена концептуальна модель даних. Вона визначає ключові сутності предметної області, їхні атрибути та зв’язки між ними.

Основні сутності системи та їхні атрибути:

– User (Користувач): id (унікальний ідентифікатор), username (ім'я користувача), email (електронна пошта), password_hash (захешований пароль), role (роль у системі: клієнт або адміністратор), created_at (дата та час реєстрації).

– GPU (Відеокарта): id (ідентифікатор), name (назва моделі), brand (компанія-виробник), architecture (графічна архітектура), memory_gb (обсяг

відеопам'яті у гігабайтах), tdp_watts (енергоспоживання у ватах), length_mm (фізична довжина у міліметрах), price (вартість), stock_count (кількість одиниць на складі).

– Order (Замовлення): id (ідентифікатор замовлення), user_id (посилання на користувача), order_date (дата оформлення), status (поточний стан виконання), total_amount (загальна сума до сплати), delivery_address (адреса доставки).

– OrderItem (Позиція замовлення): id (ідентифікатор запису), order_id (посилання на замовлення), gpu_id (посилання на товар), quantity (кількість замовлених одиниць), price_at_purchase (зафіксована ціна на момент купівлі).

– Comment (Відгук): id (ідентифікатор відгуку), gpu_id (посилання на оцінений товар), user_id (посилання на автора), text (текст коментаря), rating (числова оцінка), created_at (дата публікації).

На рисунку 2.13 наведено концептуальну модель даних, яка відображає ці сутності та базові зв'язки між ними.

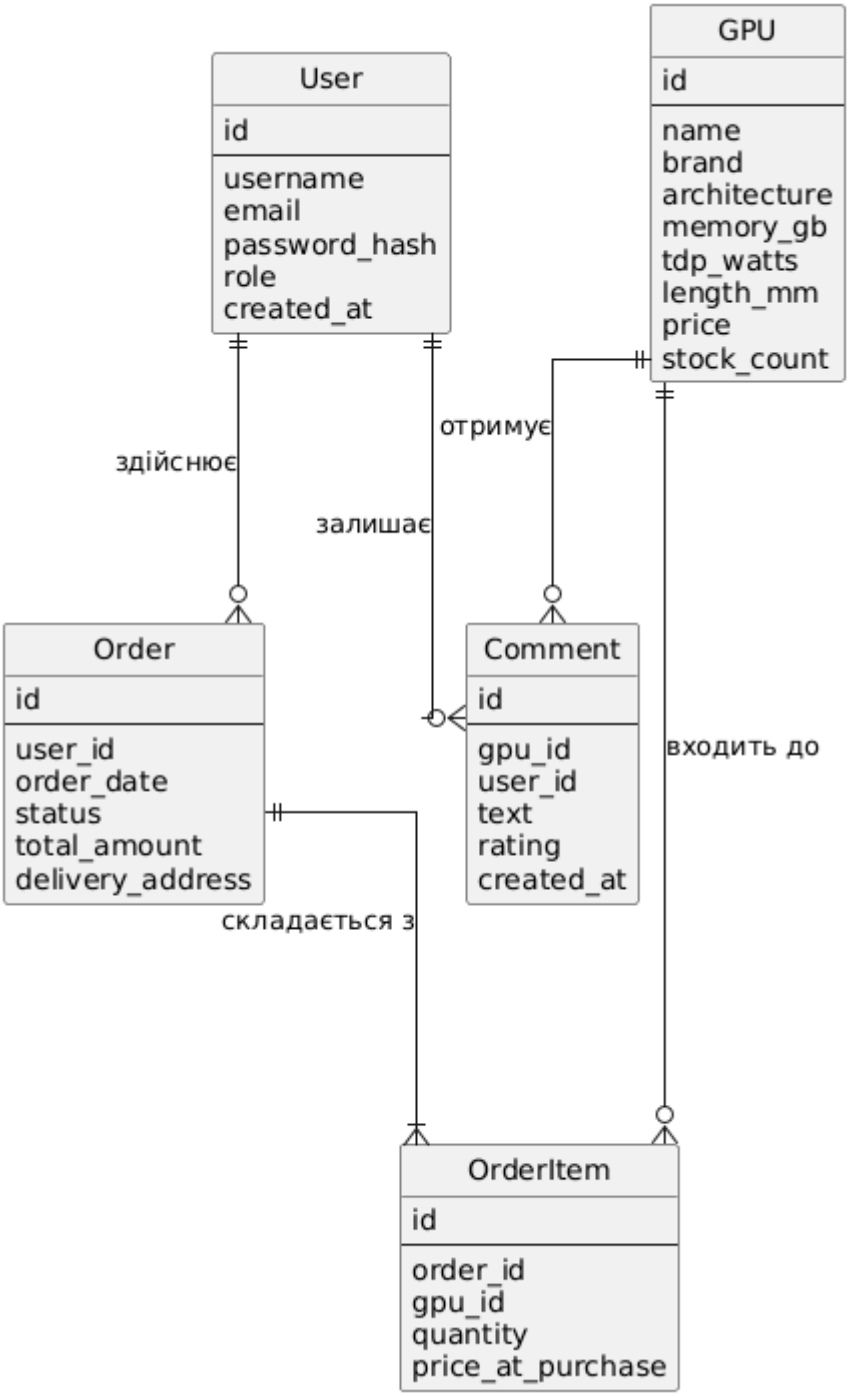


Рисунок 2.13 – Концептуальна модель даних вебзастосунку «V-Core»

Таким чином, результати ескізного проєктування створюють необхідне підґрунтя для переходу до наступного етапу – розробки технічного проєкту вебзастосунку «V-Core».

2.3.2 Технічний проєкт

Технічний проєкт деталізує внутрішню структуру системи, описуючи статичні елементи, такі як класи та їхні взаємозв'язки, а також структуру даних, що зберігаються. Цей етап є основою для подальшої реалізації програмного коду, оскільки він визначає логіку взаємодії між компонентами інтерфейсу та модулями керування станом (Context Providers).

Для представлення статичної структури вебзастосунку «V-Core» була розроблена діаграма класів, яку наведено у додатку Е. Вона ілюструє основні класи предметної області, провайдери бізнес-логіки та ієрархію сторінок і компонентів.

Центральними елементами архітектури є класи доменної моделі (пакет Domain), що описують структуру даних, та провайдери контексту (пакет Context), які реалізують логіку управління станом застосунку: від авторизації до складних розрахунків сумісності відеокарт.

Нижче наведено детальні специфікації для ключових класів, зображених на діаграмі.

Класи доменної моделі (Domain)

Клас GPU

Призначення: Основна модель даних, що описує технічні характеристики відеокарти для каталогу та системи підбору.

Атрибути:

– id: string – унікальний ідентифікатор товару.

– name, brand, series: string – назва, бренд (NVIDIA/AMD) та лінійка продукту.

– tdpWatts: number – енергоспоживання (TDP) у Ватах (ключовий параметр для перевірки сумісності).

– memoryGB: number, memoryType: string – обсяг та тип відеопам'яті.

– price: number, stockCount: number – ціна та актуальна кількість на складі.

– lengthMM: number – фізична довжина карти для перевірки сумісності з корпусом.

Клас Comment

Призначення: Модель для зберігання відгуків користувачів про конкретні моделі відеокарт.

Атрибути:

– gpuId, userId: string – прив'язка відгуку до товару та автора.

– text: string – зміст коментаря.

– rating: number? – оцінка товару за п'ятибальною шкалою.

Провайдери бізнес-логіки (Context)

Клас AuthProvider

Призначення: Керування сесіями користувачів та розмежування прав доступу (RBAC).

Методи:

– login(username, password): boolean – аутентифікація користувача.

– register(username, password): boolean – реєстрація нового клієнта.

– logout(): void – завершення поточної сесії.

– isAuthenticated, isAdmin: boolean – геттери для перевірки статусу та ролі (User/Admin).

Клас GpuProvider

Призначення: Управління даними каталогу та реалізація CRUD-операцій для адміністратора.

Методи:

– addGpu(gpu: GPU), updateGpu(gpu: GPU): void – додавання та редагування товарів.

– deleteGpu(id: string): void – видалення товару з бази даних.

– getGpuById(id: string): GPU? – отримання детальної інформації про конкретну модель.

Клас CartProvider

Призначення: Реалізація логіки кошика покупок та розрахунку фінальної вартості.

Атрибути:

– items: CartItem[] – перелік товарів, обраних клієнтом.

Методи:

– addToCart(gpu: GPU): void – додавання товару до кошика.

– totalPrice: number – автоматичний перерахунок суми замовлення.

Клас CompareProvider

Призначення: Модуль для тимчасового зберігання та порівняння технічних характеристик декількох відеокарт.

Методи:

– addToCompare(id: string), removeFromCompare(id: string): void – управління списком порівняння.

– isInCompare(id: string): boolean – перевірка наявності карти у порівняльному списку.

Для детального проєктування структури даних, яка використовується для зберігання інформації про товари, користувачів та історію продажів, концептуальну модель було перетворено на логічну, а згодом – на фізичну.

Логічна модель даних (рисунок 2.14) деталізує структуру сутностей, визначаючи точні назви атрибутів та їхні узагальнені типи (String, Integer, Decimal, DateTime). На цьому етапі фіксуються первинні (PK) та зовнішні (FK) ключі для забезпечення цілісності зв'язків між таблицями, а також встановлюються правила обов'язковості заповнення полів (NOT NULL / NULL).

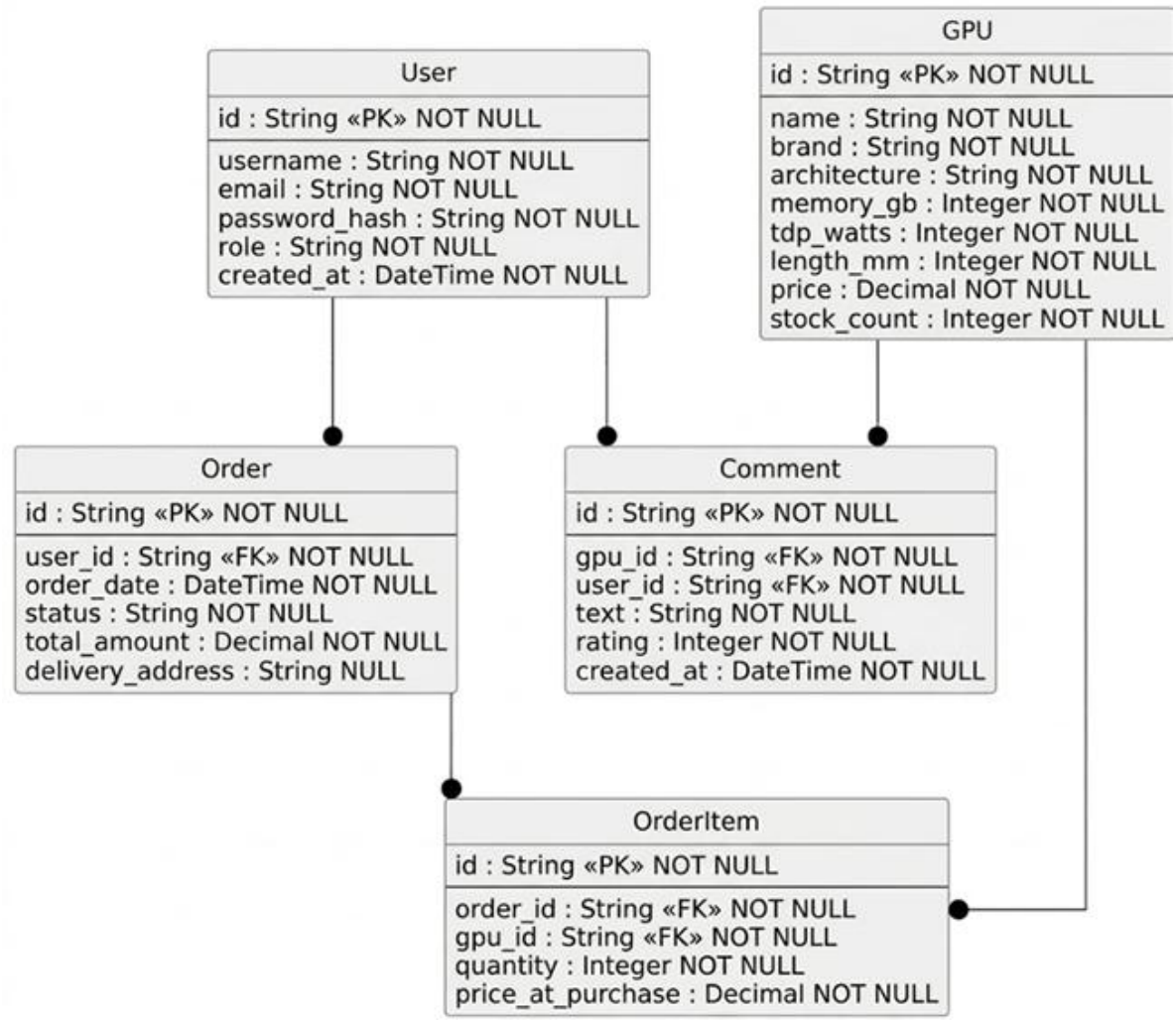


Рисунок 2.14 – Логічна модель даних вебзастосунку «V-Core»

Логічна модель приведена до третьої нормальної форми (3NF), усуває дублювання інформації, гарантує посилальну цілісність даних при виконанні CRUD-операцій та оптимізує продуктивність SQL-запитів при формуванні статистики продажів і підборі сумісних комплектуючих.

На основі логічної моделі розроблено фізичну модель (рисунок 2.15), яка адаптована під використання реляційної СУБД (наприклад, SQLite). Вона оперує специфічними SQL-типами даних, такими як VARCHAR для рядків обмеженої довжини, TEXT для великих обсягів тексту, DECIMAL(10,2) для точного збереження фінансової інформації, а також іншими типи даних, необхідними для реалізації функціональності системи.

2.3.3 Робочий проєкт

Робочий проєкт включає розробку фізичної моделі, обґрунтування вибору технологічного стеку та детальний опис реалізації основних програмних модулів системи.

Фізична модель даних:

Для реалізації бази даних була використана СУБД SQLite. Фізична модель даних наведена на рисунку 2.15.

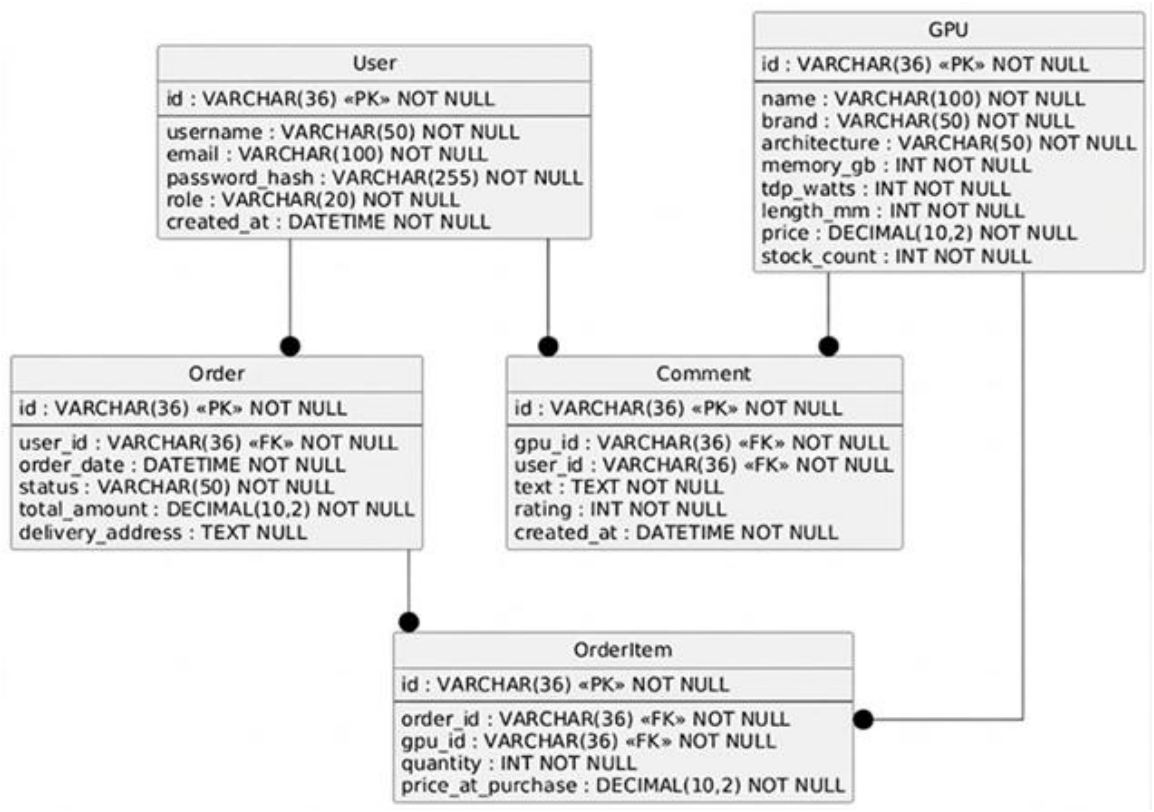


Рисунок 2.15 – Фізична модель даних вебзастосунку «V-Core»

Обґрунтування вибору засобів розробки:

Вибір технологічного стеку є фундаментальним етапом проєктування, який визначає архітектурні можливості, продуктивність, масштабованість, надійність та зручність підтримки програмного забезпечення. Для проєкту «V-Core», метою якого є створення швидкого, інтерактивного та надійного інтернет-магазину з модулем перевірки сумісності відеокарт, вибір технологій

здійснювався на основі ретельного аналізу вимог, викладених у технічному завданні. Ключовими критеріями були: висока продуктивність користувацького інтерфейсу, надійна типізація даних для уникнення помилок при розрахунках сумісності, швидкість розробки та можливість створення модульної архітектури, що легко масштабується.

Основою для побудови користувацького інтерфейсу (фронтенду) було обрано бібліотеку React (версії 18) [6]. Цей вибір обґрунтований її компонентним підходом, що дозволяє створювати перевикористовувані елементи інтерфейсу (наприклад, картки товарів, фільтри, елементи кошика). Використання технології Virtual DOM забезпечує високу продуктивність рендерингу та плавність роботи навіть при відображенні великого каталогу відеокарт.

Як основна мова програмування використовується TypeScript. На відміну від класичного JavaScript, TypeScript надає можливості суворої статичної типізації. Це критично важливо для електронної комерції: типізація структур даних (моделей відеокарт, замовлень, конфігурацій користувача) дозволяє виявляти помилки на етапі компіляції, значно підвищує надійність коду та полегшує його рефакторинг і підтримку.

Для збирання проєкту та забезпечення середовища розробки замість традиційного Webpack було обрано Vite. Це сучасний та надшвидкий інструмент збірки, який забезпечує миттєве гаряче перезавантаження модулів (HMR) під час розробки та оптимізовану компіляцію готового коду для продакшену, що суттєво прискорює робочий процес.

Для реалізації серверної частини (бекенду) та обробки бізнес-логіки використовується середовище виконання Node.js. Запуск сервера здійснюється через головний файл server.js. Це дозволяє використовувати єдину екосистему мови JavaScript/TypeScript як на клієнті, так і на сервері. Як сховище даних було обрано реляційну систему управління базами даних SQLite. Вибір SQLite обґрунтований чіткою структурою характеристик комп'ютерних комплектуючих та необхідністю забезпечення строгих зв'язків між сутностями (користувачі,

замовлення, товари), що гарантує цілісність та надійність даних (відповідність вимогам ACID).

Для забезпечення навігації та маршрутизації у застосунку типу Single Page Application (SPA) використовується стандарт індустрії – React Router DOM. Він дозволяє миттєво перемикатися між сторінками (каталог, підбір, кошик, адмін-панель) без перезавантаження браузера, зберігаючи при цьому загальний стан системи.

Особлива увага приділялася візуальному оформленню та користувацькому досвіду (UI/UX). Для стилізації було обрано утилітарний CSS-фреймворк Tailwind CSS, який дозволяє створювати адаптивні та сучасні інтерфейси безпосередньо у розмітці (JSX), усуваючи проблему конфліктів глобальних CSS-класів. У поєднанні з Tailwind використовується набір компонентів Shadcn UI (побудований на базі Radix UI), що надає високоякісні, доступні (accessible) та легко кастомізовані елементи (діалогові вікна, таблиці, селекти). Динаміка та плавність інтерфейсу забезпечується потужною бібліотекою Framer Motion, а для створення каруселей (наприклад, для промо-банерів) – гнучкою бібліотекою Embla Carousel. Векторна графіка реалізована за допомогою легкої бібліотеки іконок Lucide React. За виведення зручних спливаючих повідомлень (toasts) відповідає бібліотека Sonner, а для висувних панелей (drawers) – Vaul.

Для управління станом асинхронних запитів, кешуванням та синхронізацією даних із сервером використовується TanStack React Query. Цей інструмент ефективно вирішує проблему отримання даних з бекенду (наприклад, актуального списку відеокарт або історії замовлень), автоматично керуючи станами завантаження та помилок.

Робота з формами (авторизація, оформлення замовлення, фільтри) реалізована за допомогою React Hook Form, що мінімізує кількість перемальовувань компонентів та оптимізує продуктивність. Для безпеки та консистентності даних усі форми проходять валідацію за допомогою бібліотеки Zod, яка дозволяє декларативно описувати схеми даних та інтегрується з TypeScript.

Для аналітичного модуля адміністративної панелі, де необхідно візуалізувати статистику продажів, застосовано бібліотеку Recharts, що дозволяє будувати інтерактивні графіки та діаграми. Робота з датами та часом (форматування дати замовлень чи коментарів) оптимізована за допомогою сучасної бібліотеки date-fns.

Забезпечення якості коду є невід'ємною частиною розробки. Для написання та запуску тестів обрано фреймворк Vitest (нативна, швидка альтернатива Jest для екосистеми Vite) у поєднанні з React Testing Library для тестування UI-компонентів в умовах, максимально наближених до реального використання браузера. Стандартизація стилю коду та пошук потенційних проблем забезпечується лінтером ESLint. Допоміжні утиліти, такі як clsx, tailwind-merge та cva, використовуються для динамічного та безконфліктного управління CSS-класами при створенні варіативних компонентів.

Для управління вихідним кодом проєкту використовується система контролю версій Git, що забезпечує надійне відстеження змін та можливість розгалуження коду.

Як основне інтегроване середовище розробки (IDE) було обрано WebStorm від JetBrains. Це професійне та потужне середовище, яке має неперевершену вбудовану підтримку TypeScript, React та Tailwind CSS. Інтелектуальний аналіз коду, зручне автодоповнення, потужний рефакторинг та безшовна інтеграція з Git у WebStorm дозволили значно підвищити продуктивність та якість написання коду.

Отже, вибір кожного компонента технологічного стеку був ретельно обґрунтований з огляду на специфічні вимоги вебзастосунку «V-Core». Поєднання React, TypeScript та Vite з сучасними інструментами стилізації та управління станом дозволило створити функціональну, продуктивну, безпечну та візуально привабливу систему електронної комерції.

Опис реалізації ключових модулів:

Розробка вебзастосунку «V-Core» здійснювалася на основі модульної архітектури, що забезпечує гнучкість, розширюваність та легкість супроводу.

Кожен логічний блок системи (інтерфейс користувача, управління станом, маршрутизація, бази даних) виділено в окремий модуль, що мінімізує залежності та спрощує процес тестування. Нижче наведено опис структури проєкту та деталі реалізації ключових компонентів системи.

Загальна структура проєкту:

Структура директорій та файлів розробленого програмного забезпечення має наступний вигляд:

```
v-core-gpu-configurator-main/
├── server/                          # Бекенд (Node.js + Express)
│   ├── public/
│   │   └── uploads/                 # Завантажені зображення GPU
│   ├── database.db                  # База даних SQLite
│   ├── package.json                 # Залежності сервера
│   ├── server.js                    # API-сервер
│   └── setup.js                     # Скрипт ініціалізації БД
├── src/                             # Фронтенд (React + TS)
│   ├── api/                         # Логіка запитів до API
│   ├── components/                  # Загальні компоненти UI
│   │   ├── ui/                      # Базові UI-елементи
│   │   ├── GpuCard.tsx              # Картка товару
│   │   ├── GpuFilters.tsx           # Фільтри каталогу
│   │   └── Header.tsx               # Шапка сайту
│   ├── context/                     # React Context (Auth та ін.)
│   ├── data/                        # Статичні дані та типізація
│   ├── hooks/                       # Кастомні хуки
│   ├── pages/                       # Сторінки застосунку
│   │   ├── AdminPage.tsx            # Панель адміністратора
│   │   ├── CatalogPage.tsx          # Каталог GPU
│   │   ├── GpuDetailsPage.tsx       # Сторінка товару
│   │   └── Index.tsx                # Головна сторінка
```

```
│   ├── App.tsx                # Кореневий компонент (Routing)
│   ├── main.tsx               # Точка входу
│   └── index.css              # Глобальні стилі (Tailwind)
├── public/                    # Статика фронтенду
├── index.html                 # HTML-шаблон
├── package.json               # Залежності всього проєкту
├── tailwind.config.ts         # Конфігурація стилів
├── vite.config.ts             # Конфігурація інструменту збірки
└── tsconfig.json              # Налаштування TypeScript
```

Фронтенд: React додаток (App.tsx): Це центральний компонент, що визначає структуру маршрутизації, підключає контексти стану та глобальні сервіси.

Ініціалізація QueryClient: Використовується @tanstack/react-query для управління кешуванням та асинхронними запитами.

Контекст-провайдери: Додаток огорнуто в каскад провайдерів (Auth, Gpu, Comments, Compare, Cart), що забезпечує доступ до глобального стану з будь-якого компонента без "prop drilling".

Маршрутизація (React Router): Визначаються шляхи для всіх сторінок додатку (Головна, Каталог, Деталі GPU, Підбір, Порівняння, Кошик).

Ліниве завантаження (Lazy Loading): Сторінки завантажуються динамічно за допомогою lazy та Suspense, що зменшує початковий розмір бандла та прискорює завантаження сайту.

Глобальні UI-елементи: Підключення компонентів сповіщень (Toaster, Sonner) та підказок (TooltipProvider) для покращення UX.

App.tsx (ключові фрагменти):

```
// ... (імпорти провайдерів та сторінок) ...
const queryClient = new QueryClient();

const App = () => (
```

```
<QueryClientProvider client={queryClient}>
  <TooltipProvider>
    <AuthProvider>
      <GpuProvider>
        <CommentsProvider>
          <CompareProvider>
            <CartProvider>
              <BrowserRouter>
                <Header />
                <Suspense fallback={<PageLoader />}>
                  <Routes>
                    <Route path="/" element={<Index />} />
                    <Route path="/catalog" element={<CatalogPage />} />
                    <Route path="/gpu/:id" element={<GpuDetailsPage />} />
                    <Route path="/selector" element={<SelectorPage />} />
                    <Route path="/compare" element={<ComparePage />} />
                    <Route path="/cart" element={<CartPage />} />
                    <Route path="/login" element={<LoginPage />} />
                    <Route path="/admin" element={<AdminPage />} />
                    <Route path="*" element={<NotFound />} />
                  </Routes>
                </Suspense>
              </BrowserRouter>
            </CartProvider>
          </CompareProvider>
        </CommentsProvider>
      </GpuProvider>
    </AuthProvider>
  </TooltipProvider>
</QueryClientProvider>
```

);

Управління станом: Контексти (src/context/): Набір модулів для збереження та маніпуляції даними на рівні всього додатка.

GpuContext.tsx: Центральне сховище даних про відеокарти на клієнті. Виконує HTTP-запити до REST API сервера (/api/gpus) для завантаження актуального каталогу з бази даних SQLite. Містить функції (addGpu, updateGpu, deleteGpu) для управління каталогом, які синхронізуються з бекендом.

CartContext.tsx: Реалізує логіку кошика покупок (додавання, видалення товарів, зміна кількості, розрахунок загальної вартості за допомогою useMemo).

CompareContext.tsx: Управляє списком відеокарт, обраних користувачем для порівняння характеристик (дозволяє додавати до 4 моделей одночасно, містить перевірку дублікатів).

AuthContext.tsx: Відповідає за стан авторизації, зберігає дані про поточного користувача (адмін/клієнт) та обробляє логіку входу/виходу.

GpuContext.tsx (фрагмент ініціалізації):

```
export const GpuProvider: React.FC<{ children: React.ReactNode }> = ({ children }) => {
  const [gpus, setGpus] = useState<GPU[]>([]);

  useEffect(() => {
    fetch('/api/gpus')
      .then(res => res.json())
      .then(data => setGpus(data))
      .catch(err => console.error("Помилка завантаження даних:", err));
  }, []);

  // ... інші методи (addGpu, deleteGpu) з використанням fetch до API

  return (
    <GpuContext.Provider value={{ gpus, ... }}>
```

```
      {children}
    </GpuContext.Provider>
  );
};
```

Дані про товари (src/data/gpuData.ts): Файл забезпечує строгу типізацію даних у проєкті та містить допоміжні утиліти.

Інтерфейс GPU: Описує всі необхідні поля характеристик, гарантуючи цілісність даних при обміні між клієнтом та сервером.

```
export interface GPU {
  id: string;             // Unique identifier for the GPU
  name: string;           // Full commercial name
  brand: "NVIDIA" | "AMD";// Manufacturer
  series: string;         // Product family (e.g., RTX 40, RX 7000)
  architecture: string;   // Microarchitecture (e.g., Ada Lovelace, RDNA 3)
  memoryGB: number;       // VRAM capacity in Gigabytes
  memoryType: string;     // VRAM technology (e.g., GDDR6X)
  coreClock: number;      // Base clock speed in MHz
  boostClock: number;     // Maximum boost clock speed in MHz
  tdpWatts: number;       // Thermal Design Power in Watts
  lengthMM: number;       // Physical length for case compatibility check
  interface: string;      // Connection interface (e.g., PCIe 4.0 x16)
  outputs: string[];      // Available video outputs
  price: number;          // Price in local currency
  inStock: boolean;       // Availability status
  stockCount: number;     // Number of units available
  imageUrl?: string;      // Optional path to product image
  rating: number;         // User or benchmark rating (out of 5.0)
}
```

Функціональні сторінки:

SelectorPage.tsx: Інтерактивний "майстер" підбору GPU, який на основі відповідей користувача (бюджет, цілі використання) фільтрує базу даних та пропонує оптимальні варіанти.

CatalogPage.tsx: Сторінка зі списком усіх товарів, що підтримує сортування та багаторівневу фільтрацію за виробником, ціною та потужністю.

GpuDetailsPage.tsx: Динамічний маршрут, що відображає розширену інформацію про конкретну модель, включаючи відгуки та можливість додавання в кошик.

ComparePage.tsx: Порівняльна таблиця. Рендерить характеристики обраних відеокарт у рядки, підсвічуючи кращі значення (наприклад, менший TDP або більшу частоту).

AdminPage.tsx: Захищений інтерфейс для модерації. Дозволяє додавати нові моделі GPU через форму та редагувати існуючі ціни/наявність.

Допоміжні інструменти:

lib/utils.ts: Функція cn (classnames merge) для гнучкого управління CSS-класами Tailwind залежно від стану компонентів.

hooks/use-toast.ts: Інтерфейс для виклику спливаючих сповіщень про успішне додавання в кошик або помилки авторизації.

App.tsx: Кореневий файл, що використовує Suspense та lazy для оптимізації продуктивності шляхом розділення коду (code splitting).

Ця архітектура дозволяє легко розширювати функціонал (наприклад, додавати процесори або материнські плати), зберігаючи високу швидкість роботи завдяки React Context та клієнтському управлінню даними.

Тестування функціональності:

Для перевірки відповідності розробленого програмного забезпечення технічному завданню було проведено комплексне ручне тестування всіх ключових сценаріїв взаємодії користувача з системою:

– Навігація та фільтрація каталогу: Перевірено коректність відображення асортименту товарів, а також роботу комбінованих фільтрів (за брендом NVIDIA/AMD, обсягом відеопам'яті, архітектурою, TDP та ціновим діапазоном) і функцій сортування.

– Підбір за параметрами сумісності: Детально протестовано логіку алгоритму відсіювання несумісних відеокарт. Перевірено реакцію системи на введення користувачем різних значень потужності наявного блоку живлення (Вт) та максимальної допустимої довжини GPU (мм), включаючи обробку граничних та некоректних значень.

– Управління кошиком та оформлення замовлення: Перевірено процеси додавання товарів до кошика, зміни їхньої кількості, видалення позицій та автоматичного перерахунку загальної вартості. Протестовано форму оформлення замовлення (Checkout) на предмет коректної валідації полів (ПІБ, телефон, адреса) перед відправкою даних.

– Порівняння та відгуки: Перевірено функціонал додавання відеокарт до таблиці порівняння характеристик, а також можливість авторизованих користувачів залишати коментарі та оцінки.

– Панель адміністратора: Протестовано функції управління контентом (CRUD-операції: додавання, редагування, видалення товарів з каталогу), зміну статусу наявності, а також інструменти модерації користувацьких відгуків.

Проведене тестування підтвердило стабільність роботи компонентів інтерфейсу (React/Tailwind), коректність функціонування модулів управління станом (Context) та правильність виконання бізнес-логіки системи «V-Core».

Випробування програмного забезпечення було проведено згідно програми та методики випробувань, яка наведена в додатку В.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Результати створення програмного забезпечення

У результаті виконання кваліфікаційної роботи було розроблено програмне забезпечення «V-Core» – сучасний вебзастосунок інтернет-магазину для автоматизації онлайн-замовлень та підбору відеокарт із перевіркою апаратної сумісності. Для досягнення поставленої мети було успішно виконано завдання, які структурно відображені у відповідних розділах роботи:

– Аналіз практичної задачі інженерії програмного забезпечення: проаналізовано автоматизацію бізнес-процесів роздрібної торгівлі комп'ютерними комплектуючими. Досліджено існуючі програмні рішення (зокрема PCPartPicker, Telemart, Rozetka) та сформульовано постановку задачі на розробку власної системи, визначивши основні функціональні вимоги та обмеження.

– Проєктування та реалізація вебзастосунку: виконано комплексний аналіз вимог до програмного забезпечення з побудовою моделі та специфікацій варіантів використання (детальні специфікації наведено у додатку А). Розроблено архітектуру системи, що відображена на діаграмах компонентів та розгортання. У межах ескізного та технічного проєктів створено діаграму діяльності, інтерактивні прототипи користувацького UI-інтерфейсу, діаграму класів доменної моделі (наведено у додатку Е), а також концептуальну та логічну моделі даних. У межах підрозділу «Робочий проєкт» на основі проведеного аналізу побудовано фінальну фізичну модель даних під СУБД SQLite та повністю обґрунтовано вибір сучасного технологічного стеку: React, TypeScript, Tailwind CSS для фронтенду та Node.js для серверної частини.

– Результати розробки програмного забезпечення: продемонстровано результати розробки та проведено аналіз функціонування вебзастосунку. Описано роботу інтерактивного каталогу товарів, модуля підбору відеокарт за

параметрами системи (TDP, габарити), кошика покупок, системи авторизації та адміністративної панелі для керування контентом. Загальний обсяг розробленого програмного коду становить ~4000 рядків.

– Охорона праці: досліджено питання охорони праці при роботі з комп'ютерним обладнанням. Проаналізовано нормативні документи, структуру управління питаннями охорони праці на підприємстві та розроблено заходи щодо зменшення дії небезпечних і шкідливих факторів у відділі розробки та тестування програмного забезпечення.

Також розроблено повний комплект супровідної документації та додаткових графічних і технічних матеріалів, що представлені у додатках роботи:

– Додаток А: Технічне завдання на розробку ПЗ.
– Додаток Б: Опис програмного забезпечення.
– Додаток В: Програма та методика випробувань.
– Додаток Г: Керівництво користувача (оператора).
– Додаток Д: Лістинги ключових програмних модулів.
– Додаток Е: Діаграма класів вебзастосунку «V-Core».

3.2 Аналіз функціонування програмного забезпечення

Для перевірки працездатності та демонстрації функціональних можливостей розробленого вебзастосунку магазину «V-Core» було проведено наскрізне тестування основних користувацьких сценаріїв (End-to-End тестування). Нижче наведено покрокову демонстрацію роботи системи для ролей звичайного клієнта та адміністратора.

А. Реєстрація нового користувача

Процес створення облікового запису є важливим кроком для отримання можливості відстежувати історію замовлень та залишати відгуки.

– Крок 1: Користувач відкриває головну сторінку та переходить до розділу авторизації/реєстрації. Система відображає форму для введення облікових даних (рисунок 3.1).

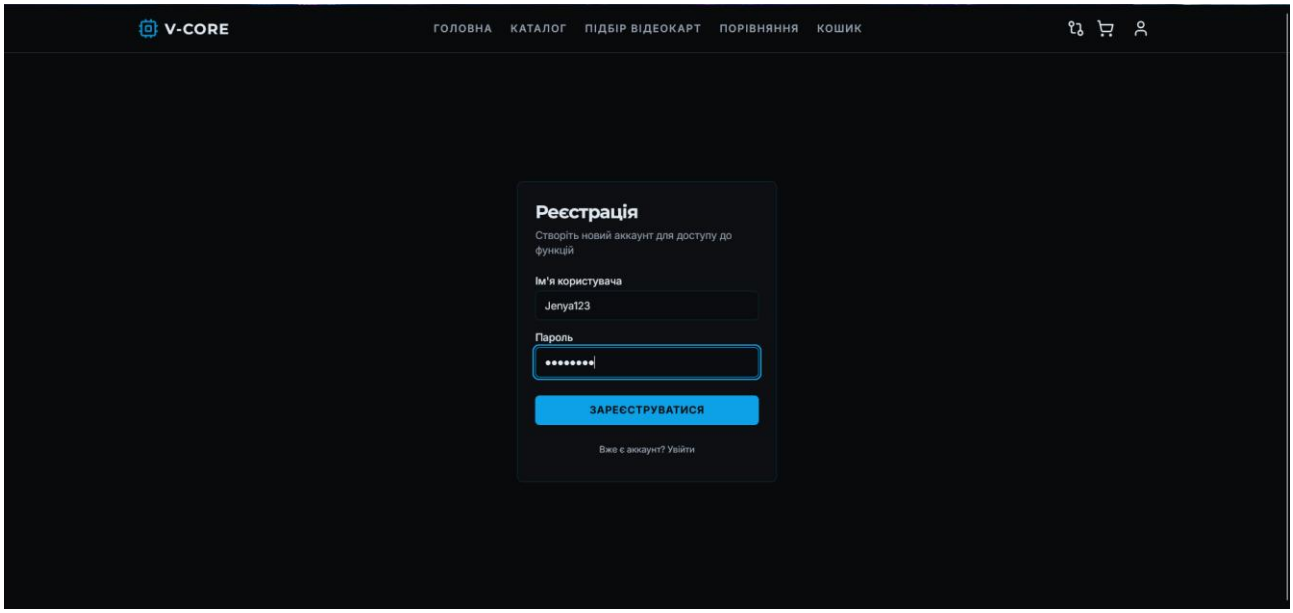


Рисунок 3.1 – Інтерфейс форми реєстрації нового користувача

– Крок 2: Клієнт заповнює необхідні поля: ім'я, електронна пошта та пароль. Завдяки бібліотеці Zod система виконує миттєву валідацію введених даних (перевірка формату email, складності пароля) ще до відправки запиту на сервер.

– Крок 3: Після натискання кнопки підтвердження сервер обробляє запит, хешує пароль та повертає токен доступу. Користувач отримує системне сповіщення (toast) про успішну реєстрацію та автоматично перенаправляється в особистий кабінет (рисунок 3.2).

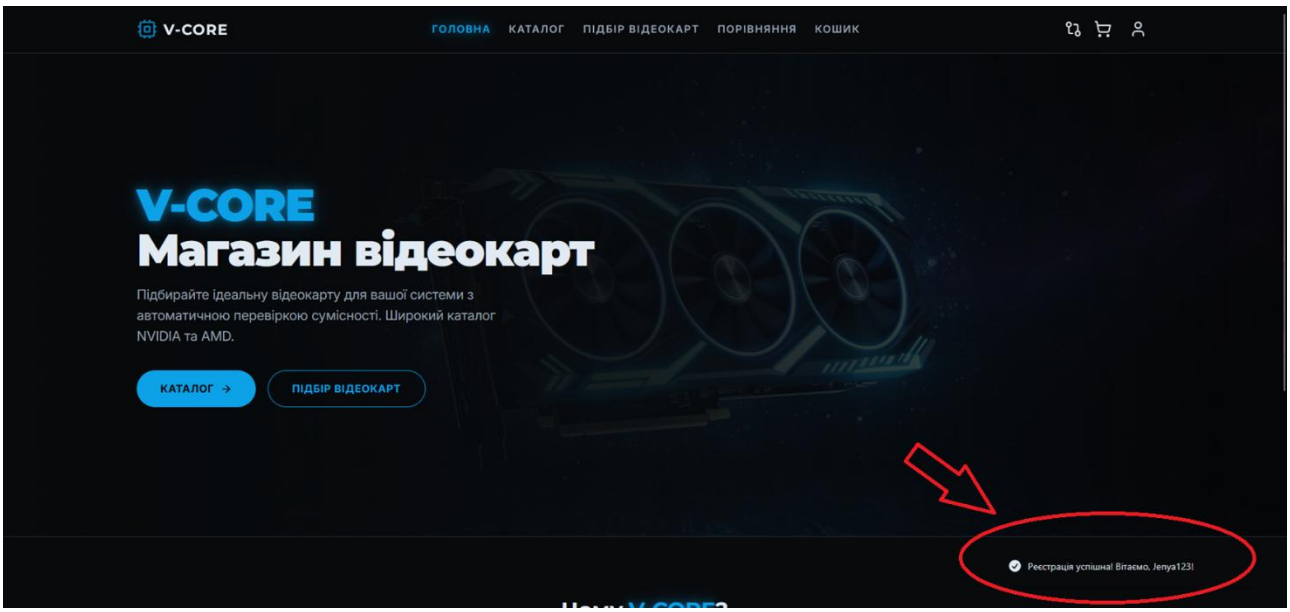


Рисунок 3.2 – Успішне завершення реєстрації

Б. Підбір та оформлення замовлення відеокарти

Це ключовий бізнес-процес системи, який включає роботу унікального алгоритму перевірки апаратної сумісності.

– Крок 1: Клієнт переходить до розділу «Підбір відеокарт» та вказує параметри свого системного блоку, наприклад: потужність наявного блоку живлення – 650 Вт, максимальна допустима довжина GPU – 300 мм (рисунок 3.3).

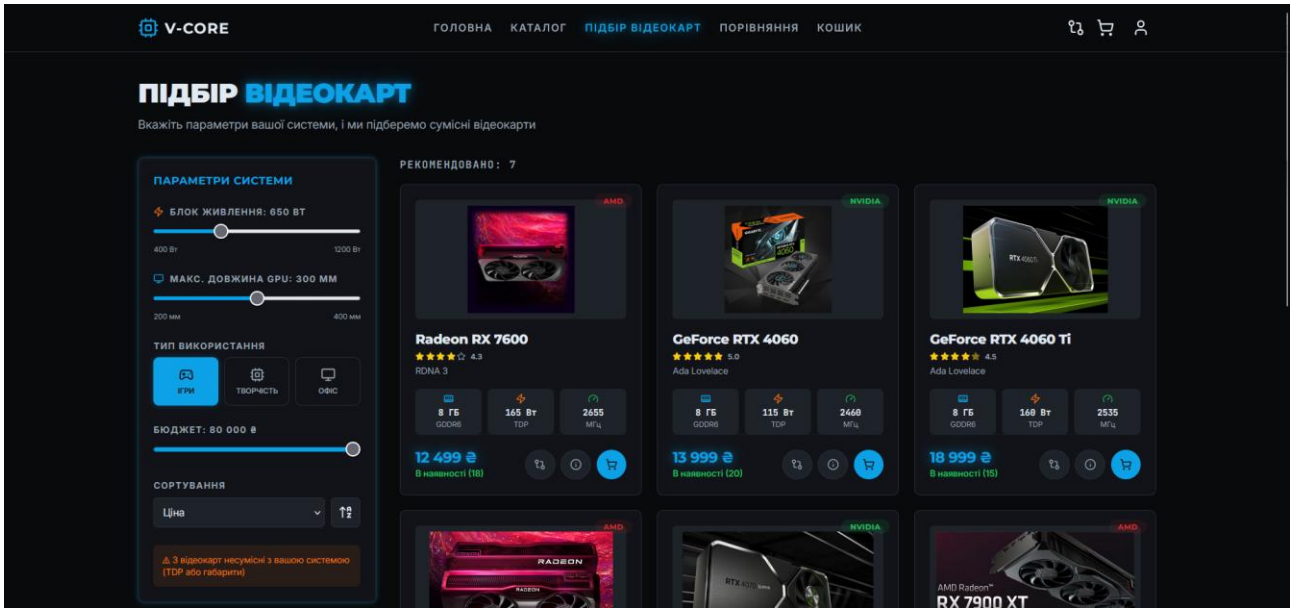


Рисунок 3.3 – Введення параметрів системи для підбору

– Крок 2: Система динамічно відфільтровує каталог товарів, залишаючи лише ті відеокарти, енергоспоживання (TDP) та габарити яких технічно відповідають введеним параметрам. Користувач обирає оптимальну модель та натискає «Додати в кошик».

– Крок 3: Клієнт переходить до кошика (рисунок 3.4), де має змогу змінити кількість товарів або видалити позиції. Система автоматично перераховує загальну вартість замовлення.

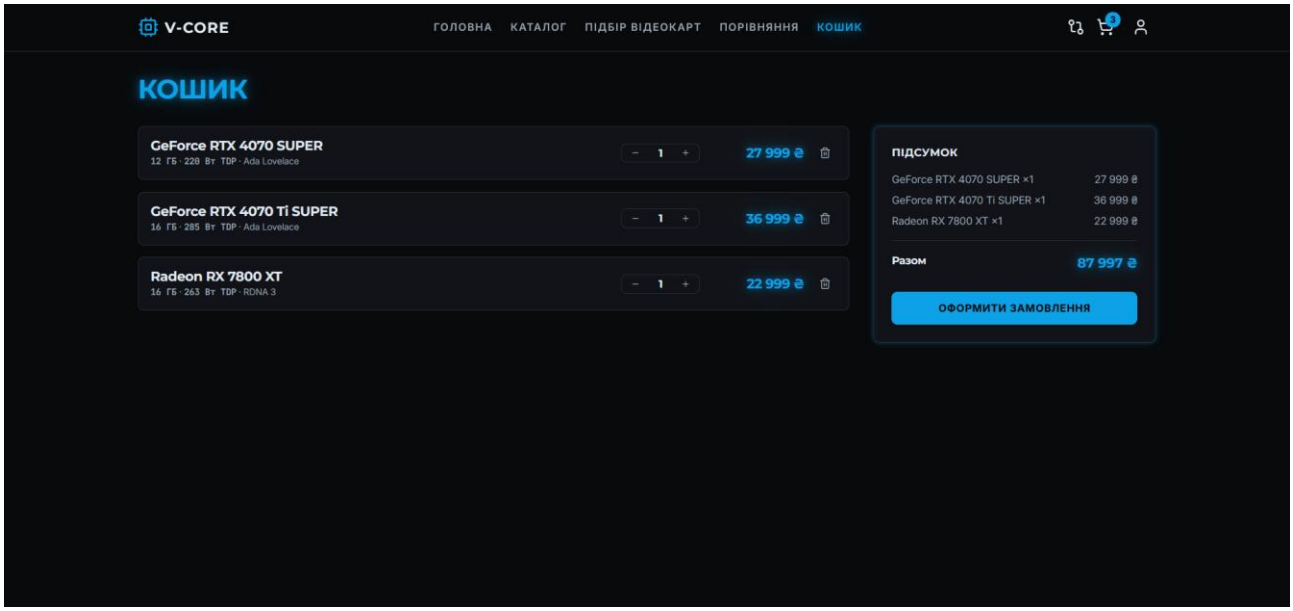


Рисунок 3.4 – Інтерфейс кошика покупок

– Крок 4: Після натискання «Оформити замовлення» користувач потрапляє на сторінку Checkout. Він заповнює контактні дані, адресу доставки та підтверджує покупку (рисунок 3.5). Дані зберігаються у базі даних SQLite, а залишок товару на складі автоматично зменшується.

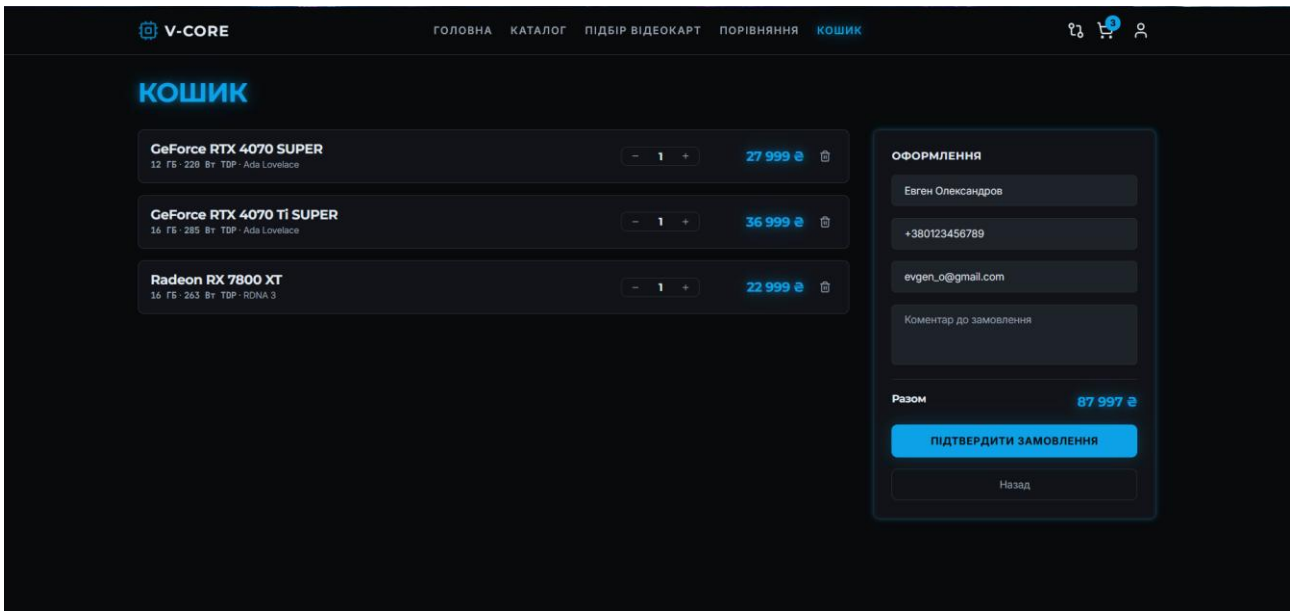


Рисунок 3.5 – Форма оформлення замовлення

В. Робота в панелі адміністратора

Для управління контентом магазину та обробки замовлень реалізовано захищений модуль адміністратора.

– Крок 1 (Авторизація): Співробітник переходить на сторінку входу та вводить службові облікові дані (для тестування: Логін – admin, Пароль – admin). Система перевіряє права доступу на основі ролей (RBAC) та надає доступ до закритої частини сайту (рисунок 3.6).

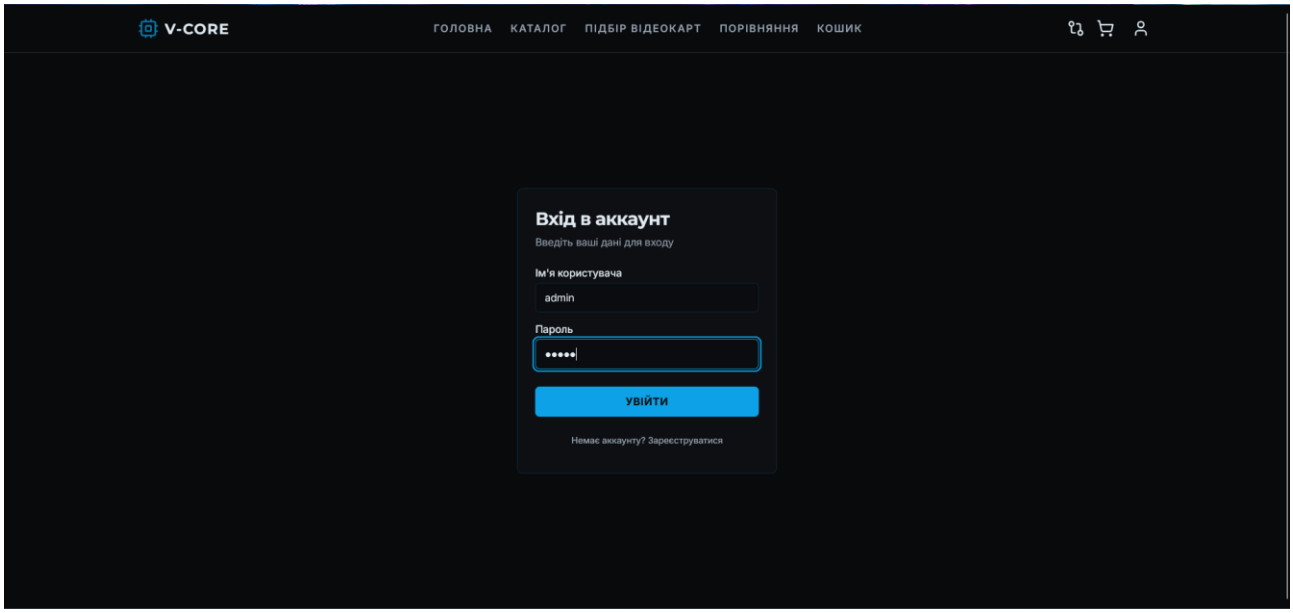


Рисунок 3.6 – Авторизація адміністратора системи

– Крок 2 (Управління каталогом): В розділі «Адмін-панель» адміністратор бачить таблицю всіх відеокарт. При натисканні кнопки додавання або редагування відкривається форма, де заповнюються технічні характеристики: назва, бренд, архітектура, обсяг пам'яті, TDP, ціна та кількість на складі (рисунок 3.7).

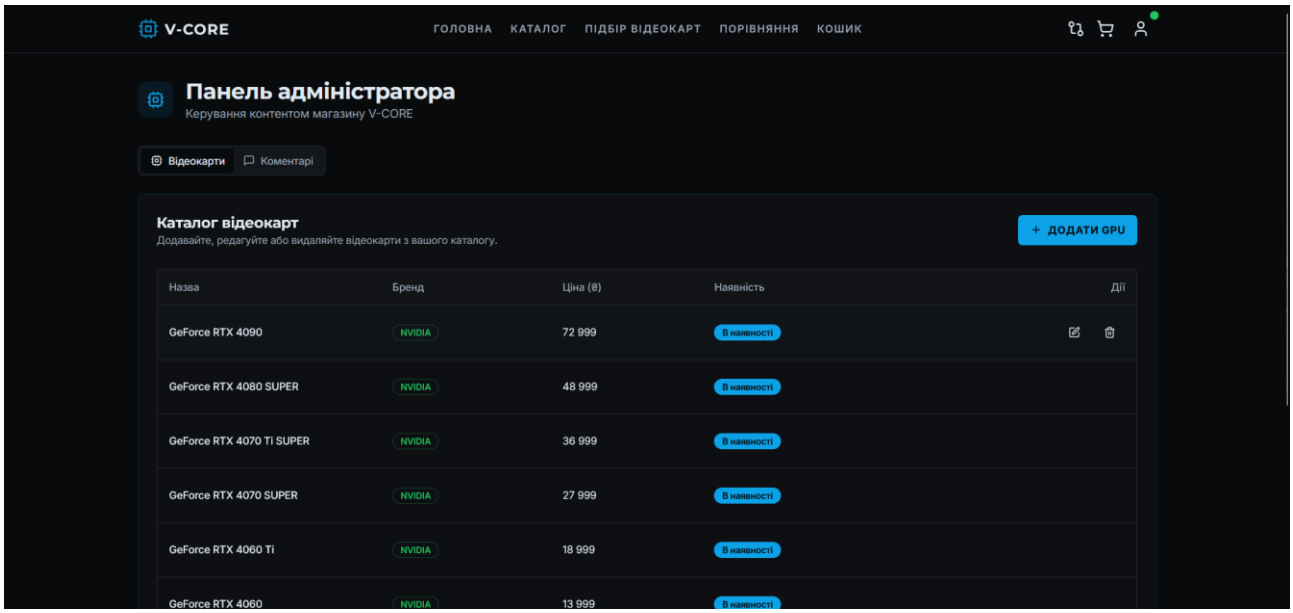


Рисунок 3.7 – Панель адміністратора: керування асортиментом товарів

– Крок 3 (Модерація): В окремій вкладці реалізовано модуль модерації коментарів. Адміністратор переглядає нові відгуки користувачів та має можливість видалити коментар, якщо він містить спам або порушує правила платформи (рисунок 3.8).

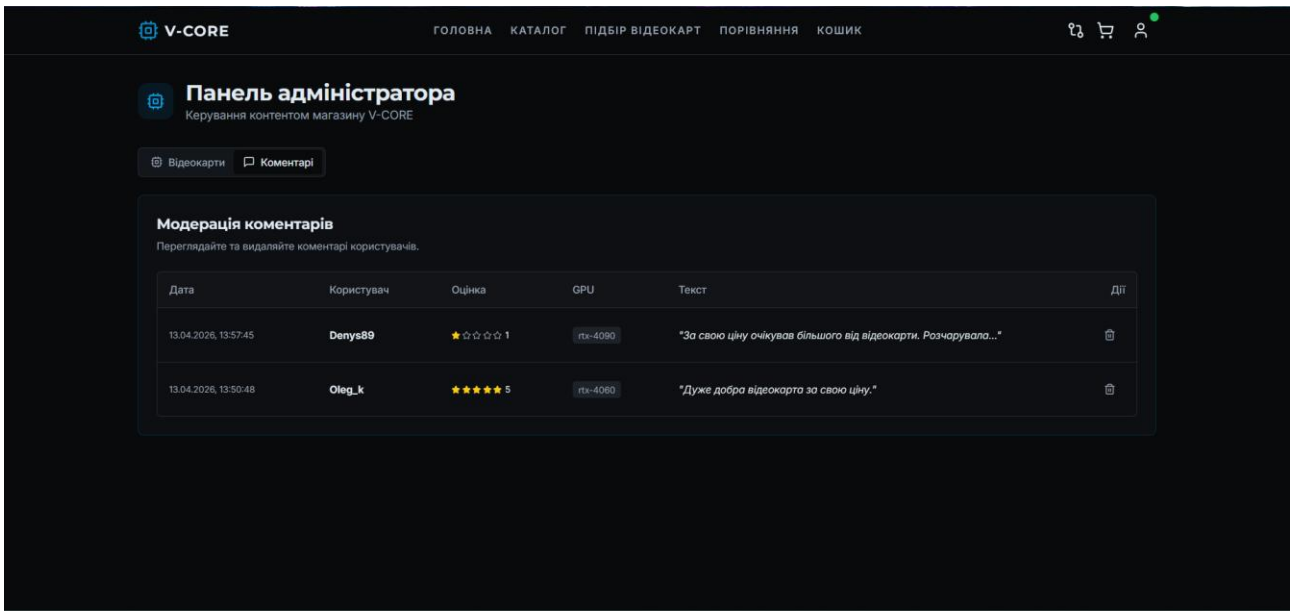


Рисунок 3.8 – Панель адміністратора: модерація відгуків

Висновки щодо аналізу функціонування:

Проведена демонстрація підтверджує, що розроблений вебзастосунок «V-Core» успішно та безпомилково виконує всі заявлені в технічному завданні функції. Інтерфейс, побудований на базі React та Tailwind CSS, забезпечує високу швидкість відгуку та зручність взаємодії. Валідація даних працює коректно як на клієнтській, так і на серверній стороні. Ключова функція – алгоритм підбору – надійно відсіює комплектуючі, що не підходять за параметрами енергоспоживання або габаритами, що робить платформу зручним та безпечним інструментом для здійснення покупок.

4 ОХОРОНА ПРАЦІ

4.1 Основні законодавчі і нормативні документи, які регламентують охорону праці в Україні

Охорона праці в Україні є важливою складовою державної політики, спрямованої на забезпечення безпечних та здорових умов праці для всіх працівників, незалежно від сфери діяльності підприємства, у тому числі в галузі розробки програмного забезпечення та торгівлі комп'ютерною технікою. Законодавча та нормативна база, що регулює охорону праці, включає низку документів, які визначають права та обов'язки роботодавців і працівників, а також стандарти безпеки на робочих місцях. Основними законодавчими та нормативними документами, які регламентують охорону праці в Україні, є:

– Конституція України: Стаття 43 гарантує кожному право на належні, безпечні та здорові умови праці, а також на соціальний захист у разі втрати працездатності [7].

– Кодекс законів про працю України (КЗпП): Головний законодавчий акт, що регулює трудові відносини в Україні. Розділ XI КЗпП присвячений охороні праці та визначає обов'язки роботодавців щодо забезпечення безпеки праці та права працівників на охорону праці [8].

– Закон України "Про охорону праці": Основний закон, що визначає правові, економічні та соціальні засади охорони праці в Україні. Закон зобов'язує роботодавців створювати безпечні умови праці, проводити навчання та інструктажі з питань охорони праці, забезпечувати засобами індивідуального захисту тощо [9].

– Закон України "Про загальнообов'язкове державне соціальне страхування": Визначає порядок страхування працівників від нещасних випадків на виробництві та професійних захворювань, порядок надання страхових виплат і соціальних послуг [10].

– Державні нормативно-правові акти з охорони праці (ДНАОП): Нормативні акти, що містять конкретні вимоги та правила щодо безпеки праці в різних галузях. ДНАОП розробляються Міністерством соціальної політики та іншими органами виконавчої влади.

– Гігієнічні нормативи та санітарні правила: Встановлюють вимоги до санітарно-гігієнічних умов на робочих місцях (зокрема, для робочих місць, обладнаних персональними комп'ютерами), включаючи рівні шуму, освітленості, мікроклімату, електромагнітного випромінювання.

– Стандарти безпеки праці (ДСТУ): Національні стандарти, що регулюють безпеку праці, розробляються Державним комітетом України з питань технічного регулювання та споживчої політики.

– Інструкції з охорони праці: Локальні нормативні документи, які розробляються безпосередньо на підприємствах і містять конкретні вимоги щодо безпечного виконання робіт для різних професій (наприклад, інструкція з охорони праці для розробника програмного забезпечення, системного адміністратора або менеджера магазину).

– Положення про систему управління охороною праці (СУОП): Визначає організаційні основи системного управління охороною праці на підприємствах, установах та організаціях.

Дотримання цих законодавчих та нормативних документів забезпечує належний рівень безпеки на робочих місцях, зменшує ризики виробничого травматизму та професійних захворювань, а також сприяє створенню здорових та безпечних умов праці для всіх працівників магазину «V-Core» та IT-спеціалістів, залучених до розробки програмного забезпечення.

4.2 Структура управління питаннями охорони праці на підприємстві «V-Core»

Управління питаннями охорони праці в магазині «V-Core» є важливою складовою загальної системи управління підприємством. Основною метою

системи управління охороною праці є забезпечення безпечних та здорових умов праці для всіх працівників магазину, запобігання виробничому травматизму та професійним захворюванням. Структура управління питаннями охорони праці на підприємстві включає кілька ключових елементів та рівнів відповідальності, які діють узгоджено для досягнення комплексного результату.

Однією з визначальних рис цієї структури є чіткий розподіл функцій між керівництвом та персоналом, що дозволяє ефективно взаємодіяти в процесі планування, організації та здійснення заходів із охорони праці. Крім того, завдяки постійному моніторингу за дотриманням встановлених вимог безпеки вдається оперативно реагувати на будь-які порушення і запобігати можливим ризикам. Нижче наведено основні посади та обов’язки працівників, які відповідають за реалізацію системи управління охороною праці в магазині «V-Core».

Директор магазину:

– несе загальну відповідальність за стан охорони праці на підприємстві;

– забезпечує фінансування заходів з охорони праці;

– приймає рішення про призначення відповідальних осіб за охорону праці;

– контролює виконання законодавчих і нормативних вимог з охорони праці.

Інженер з охорони праці:

– відповідає за розробку, впровадження та контроль виконання заходів з охорони праці;

– проводить регулярні перевірки умов праці та виявлення потенційних ризиків;

– організовує навчання та інструктажі з охорони праці для працівників;

– забезпечує ведення документації з охорони праці.

Діяльність інженера з охорони праці значною мірою визначає, наскільки ефективно у магазині дотримуються встановлених правил безпеки. За допомогою регулярних перевірок та аудиту робочих місць він виявляє можливі недоліки в організації праці й оперативно розробляє рекомендації для їх усунення. Організовані ним інструктажі та навчання формують у персоналу розуміння важливості дотримання вимог охорони праці і сприяють загальному підвищенню культури безпеки в колективі.

Співробітники:

– зобов'язані дотримуватися вимог охорони праці, встановлених на підприємстві;

– виконують інструкції та правила безпеки під час роботи;

– повідомляють керівництво про будь-які небезпечні умови праці або нещасні випадки.

Навчання та інструктажі:

– всі працівники проходять первинний, повторний, позаплановий та цільовий інструктажі з охорони праці;

– проводяться регулярні навчання та тренування з питань охорони праці, евакуації при пожежі та надання першої медичної допомоги.

Організація навчальних заходів має систематичний характер і спрямована на підтримку належного рівня обізнаності персоналу щодо вимог безпеки. Проведення тренувань із евакуації та надання першої медичної допомоги забезпечує готовність працівників до оперативних дій у разі надзвичайних ситуацій. Такий підхід допомагає мінімізувати ризики для життя та здоров’я, а також гарантує чітку координацію дій у стресових обставинах.

Документація з охорони праці:

– ведеться облік всіх заходів з охорони праці, включаючи журнали інструктажів, акти перевірок, плани заходів з покращення умов праці;

– розробляються локальні нормативні акти, положення та інструкції з охорони праці, які відповідають законодавчим вимогам.

Структура управління охороною праці в магазині «V-Core» забезпечує комплексний підхід до створення безпечних умов праці, з чітко визначеними відповідальностями на всіх рівнях управління, що сприяє підвищенню рівня безпеки та захисту здоров'я працівників.

4.3 Розробка заходів по зменшенню дії небезпечних і шкідливих факторів на підприємстві «V-Core»

У відділі розробки програмного забезпечення та тестування магазину «V-Core» найбільш характерними шкідливими та небезпечними виробничими факторами є тривала робота за комп’ютером, підвищені вимоги до зору, електричні ризики при роботі з офісною технікою, а також необхідність дотримання правил безпеки при поводженні з апаратним забезпеченням (відеокартами, блоками живлення, системними блоками), яке може знадобитися для тестування систем підбору чи демонстрації в процесі роботи.

Для створення безпечних і комфортних умов праці слід упроваджувати як організаційні, так і технічні заходи.

Забезпечення ергономічних умов праці:

– Монітори мають відповідати сучасним стандартам із частотою розгортки не менше ніж 75 Гц (рекомендовано від 100 Гц) та технологіями захисту зору (Flicker-Free, Low Blue Light), щоб зменшити навантаження на очі.

– Оптимальні мікрокліматичні показники: Рекомендовано підтримувати температуру в робочому приміщенні на рівні 23 °C у теплу пору року та близько 18 °C у холодну, з обов’язковим провітрюванням приміщень.

– Ергономіка робочого місця: Регульовані столи й крісла (з підтримкою попереку та можливістю налаштування висоти). Верхній край монітора слід розташовувати на рівні очей або трохи нижче.

– Дотримання санітарно-гігієнічних норм:

- Вологе прибирання: Рекомендується проводити щодня, щоб зменшити кількість пилу (який є критично шкідливим для комп'ютерної техніки) та підтримувати чистоту приміщення.

- Кондиціонування й вентиляція: Встановлення кондиціонера або системи вентиляції для підтримання оптимальної температури й вологості повітря.

- Освітлення: У темний час доби варто використовувати люмінесцентні або LED-лампи з високим коефіцієнтом світловіддачі (75+ лм/Вт) і тривалим строком служби (понад 10 000 год).

– Дотримання правил поводження з комп'ютерними комплектуючими:

- Безпечне зберігання: Комп'ютерне обладнання (відеокарти, материнські плати, тощо), яке може перебувати у відділі розробки ПЗ для тестування, повинне зберігатися в окремому приміщенні чи на стелажах із забезпеченням належних умов (захист від статичної електрики, сухість, відповідний температурний режим).

- Правила користування: Співробітники, які працюють із залізом для тестових стендів, мають проходити короткий інструктаж щодо правил збірки/розбірки, використання антистатичних браслетів та уникнення травмонебезпечних ситуацій (наприклад, порізи від гострих країв радіаторів чи корпусів, підйом важких джерел безперебійного живлення).

- Контроль стану та справності: Рекомендується періодично перевіряти тестове обладнання на наявність пошкоджень (оголені кабелі, оплавлені конектори, пошкоджена ізоляція), аби уникнути короткого замикання.

Електробезпека та протипожежні заходи:

– Пристрої заземлення: Необхідно забезпечити якісне заземлення комп’ютерів, тестових стендів та периферійних пристроїв.

– Захисна автоматика: Використання пристроїв захисного відключення (ПЗВ), джерел безперебійного живлення (ДБЖ) і мережевих

фільтрів знижує ризик короткого замикання й пошкодження дороговартісної техніки.

– Протипожежні засоби: Рекомендується встановити вуглекислотні або порошкові вогнегасники (оптимальні для гасіння електроприладів), димові датчики та підлогу з негорючих матеріалів, а також розмістити план евакуації на видному місці.

Організація робочого часу та профілактика перевтоми:

– Регулярні перерви: Слід робити короткі перерви кожні 1–2 години, аби зменшити навантаження на очі та м’язи (гімнастика для рук і шиї).

– Планування завдань: Використання систем управління проєктами (трекерів завдань, календарів) допомагає рівномірно розподіляти навантаження та мінімізувати стрес під час розробки програмного забезпечення.

– Психологічний клімат: Підтримання дружньої атмосфери в колективі, проведення мотиваційних заходів та тренінгів із боротьби зі стресом.

Навчання й контроль з питань охорони праці:

– Інструктажі: Регулярні вступні та повторні інструктажі з охорони праці, включно з правилами роботи за комп’ютером та безпечного поводження з електроприладами і комплектуючими.

– Планові перевірки: Періодичний огляд робочих місць, перевірка стану офісної та комп'ютерної техніки, аналіз зауважень працівників.

– Документація: Розробка й оновлення локальних інструкцій, актів і журналів реєстрації інцидентів, що забезпечує системний підхід до охорони праці.

Упровадження наведених заходів у відділі розробки програмного забезпечення «V-Core» сприяє зменшенню впливу небезпечних і шкідливих факторів, а також підвищує ефективність і безпеку праці співробітників, створюючи комфортне робоче середовище навіть за умови постійної роботи з різноманітним комп'ютерним обладнанням.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було успішно досягнуто поставленої мети – розроблено вебзастосунок магазину «V-Core», який автоматизує процеси онлайн-замовлень та надає інтелектуальний інструментарій для підбору відеокарт із перевіркою апаратної сумісності.

У процесі роботи було вирішено наступні завдання:

– Проаналізовано предметну область, зокрема особливості ринку комп'ютерних комплектуючих та технічні критерії сумісності відеокарт із компонентами ПК (енергоспоживання TDP, фізичні габарити, вимоги до блоків живлення). Аналіз існуючих аналогів (PCPartPicker, Telemart та ін.) підтвердив актуальність створення вузькоспеціалізованого рішення, що мінімізує ризики помилкового вибору обладнання користувачем.

– Сформульовано детальні функціональні та нефункціональні вимоги до системи «V-Core» та розроблено технічне завдання, що стало фундаментом для подальшого проєктування та реалізації.

– Спроєктовано архітектуру вебзастосунку на основі сучасного технологічного стеку (React, Node.js, SQLite, Tailwind CSS). Обрана модульна структура забезпечує високу швидкість рендерингу інтерфейсу на стороні клієнта та надійність обробки даних на сервері.

– Реалізовано програмний продукт «V-Core», який включає інтерактивний каталог товарів із багатофакторною фільтрацією, модуль інтелектуального підбору за параметрами системи користувача, систему управління кошиком та замовленнями, а також захищену панель адміністратора для керування контентом.

– Проведено комплексне тестування розробленого програмного забезпечення. Функціональне тестування підтвердило коректність роботи всіх компонентів системи. Випробування алгоритму сумісності виявили наступні закономірності:

- Впровадження автоматичного розрахунку необхідної потужності БЖ (із урахуванням запасу на систему) дозволяє на 100% виключити відображення моделей GPU, що перевищують ліміти живлення користувача.

- Перевірка геометричних параметрів (довжини GPU) ефективно відсіює комплектуючі, що фізично не сумісні з обмеженнями корпусу ПК, що значно підвищує якість користувацького досвіду (UX).

- Реактивний підхід до розробки інтерфейсу (React) забезпечує миттєвий відгук системи на зміну параметрів фільтрації без перезавантаження сторінок.

– Розроблено повний комплект програмної документації, що включає технічне завдання, опис програмного забезпечення, програму та методику випробувань, керівництво користувача, а також підготовлено необхідні графічні матеріали (діаграму класів) та лістинги вихідного коду.

– Проаналізовано питання охорони праці, пов’язані з розробкою та експлуатацією програмного продукту, визначено заходи для створення безпечних умов праці розробника та кінцевих користувачів.

Таким чином, кваліфікаційна робота охопила повний цикл розробки програмного забезпечення: від аналізу вимог та проєктування архітектури до програмної реалізації, тестування та документування. Створений вебзастосунок «V-Core» є завершеним комерційно орієнтованим продуктом, готовим до впровадження у сфері електронної комерції для автоматизації продажів комп’ютерної техніки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ринок електронної комерції в Україні: тенденції та перспективи розвитку в умовах сучасності. URL: https://ain.ua/ (дата звернення: 20.03.2026)

2. PCPartPicker – Pick parts. Build your PC. Compare and share. URL: https://pcpartpicker.com/ (дата звернення: 20.03.2026)

3. Інтернет-магазин комп'ютерної техніки та комплектуючих TELEMART.UA. URL: https://telemart.ua/ (дата звернення: 20.03.2026)

4. Інтернет-магазин комп'ютерних комплектуючих – ROZETKA. URL: https://rozetka.com.ua/ua/computers-notebooks/c80253/ (дата звернення: 20.03.2026)

5. FASTPANEL – Simple and powerful server management panel. URL: https://fastpanel.direct/ (дата звернення: 21.03.2026)

6. React – Офіційна документація бібліотеки JavaScript для створення користувацьких інтерфейсів. URL: https://react.dev/ (дата звернення: 21.03.2026)

7. Конституція України: Закон України від 28.06.1996 № 254к/96-ВР. URL: https://zakon.rada.gov.ua/laws/show/254%D0%BA/96-%D0%B2%D1%80#Text (дата звернення: 02.06.2026)

8. Конституція України. URL: https://zakon.rada.gov.ua/laws/show/254к/96-вр#Text (дата звернення: 22.03.2026)

9. Кодекс законів про працю України. URL: https://zakon.rada.gov.ua/laws/show/322-08#Text (дата звернення: 22.03.2026)

10. Закон України "Про охорону праці". URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 22.03.2026)

11. Закон України "Про загальнообов’язкове державне соціальне страхування". URL: https://zakon.rada.gov.ua/laws/show/1105-14#Text (дата звернення: 22.03.2026)

ДОДАТОК А – Технічне завдання на розробку вебзастосунку для автоматизації обробки онлайн-замовлень та підбору відеокарт для магазину «V-Core»

Вступ

Повна назва продукту: вебзастосунок для автоматизації обробки онлайн-замовлень та підбору відеокарт для магазину «V-Core». Область застосування – магазин «V-Core».

1 Підстава для розробки

Підставою для розробки даного програмного забезпечення є наказ на затвердження тем кваліфікаційних робіт бакалаврів зі спеціальності 121 «Інженерія програмного забезпечення» в Національному університеті кораблебудування імені адмірала Макарова.

2 Призначення розробки

2.1 Експлуатаційне призначення

Експлуатаційне призначення програмного забезпечення для автоматизації обробки онлайн-замовлень та підбору відеокарт для магазину «V-Core» полягає в автоматизації та оптимізації процесів обліку товарів у каталозі, оформлення продажів, управління складськими залишками та взаємодії з клієнтами. Це забезпечує ефективне управління ресурсами магазину, знижує ймовірність помилок при підборі технічно сумісного обладнання, підвищує швидкість обробки замовлень та полегшує роботу персоналу, роблячи процес купівлі зручнішим для клієнтів.

2.2 Функціональне призначення

Функціональне призначення полягає в можливості користувачів виконувати такі функції:

– реєстрація та авторизація користувачів;

– перегляд та обробка інформації про товари в каталозі (додавання, видалення, редагування технічних характеристик);

– перегляд та обробка інформації про клієнтів (додавання, видалення, редагування);

– підбір та фільтрація відеокарт за критичними параметрами (архітектура GPU, TDP, обсяг пам'яті тощо);

– створення замовлення на покупку (додавання усієї інформації щодо продажу, пошук товару та клієнта, внесення нового клієнта у разі його відсутності);

– редагування та видалення існуючого замовлення з відповідним поверненням товару на баланс складу;

– оновлення статусу замовлення (наприклад, на «виконано» або «відправлено») для подальшого відстеження;

– перегляд списку усіх товарів, доданих у конкретне замовлення клієнта;

– перегляд та видалення зареєстрованих співробітників (менеджерів);

– вихід з облікового запису.

3 Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Додатком будуть користуватися три основні групи користувачів: адміністратор (admin) та менеджер (moderator) в адміністративній панелі, а також клієнт (customer) через публічну вітрину магазину. В адміністративній панелі менеджер може виконувати усі функції адміністратора крім:

– додавання та редагування інформації про товари в каталозі (тільки адміністратор);

– перегляд та видалення зареєстрованих менеджерів (тільки адміністратор).

Нижче всі функції, які доступні адміністратору:

– реєстрація співробітників;

Вхідні дані: логін, пароль, роль.

Результат: створення нового облікового запису в системі.

– авторизація користувачів;

Вхідні дані: логін, пароль, роль (підтягується автоматично згідно з даними вже зареєстрованого користувача).

Результат: вхід у систему з правами доступу ролі, яка була визначена під час реєстрації (admin/ moderator).

– обробка інформації про товари на складі (додавання, видалення, редагування);

Вхідні дані: артикул товару, фото, назва, технічні характеристики (архітектура GPU, обсяг пам'яті, шина, TDP, інтерфейси), кількість, ціна.

Результат: відображення, додавання, видалення або редагування товарів у каталозі (додавання/редагування доступне тільки admin).

– обробка інформації про клієнтів (додавання, видалення, редагування);

Вхідні дані: прізвище, ім'я, телефон, електронна пошта, адреса доставки.

Результат: додавання, видалення або редагування даних клієнтів.

– підбір та фільтрація відеокарт;

Вхідні дані: параметри фільтрації (бажаний обсяг пам'яті, ліміт TDP блоку живлення, виробник, діапазон цін).

Результат: відображення повного списку відеокарт, відображення списку відеокарт, що відповідають заданим критеріям сумісності та вимогам клієнта.

– створення замовлення на покупку (додавання усієї інформації щодо продажу, пошук товару та клієнта, внесення нового клієнта у разі його відсутності);

Вхідні дані: номер замовлення, ПІБ клієнта, дата замовлення, статус оплати, менеджер, що оформлював покупку, спосіб доставки, перелік товарів, загальна сума.

Результат: створення нового запису продажу зі списанням товару зі складу.

– редагування та видалення замовлення;

Вхідні дані: номер замовлення, ПІБ клієнта, дата замовлення, статус оплати, перелік товарів, загальна сума.

Результат: редагування та видалення замовлення з відповідним поверненням товару на баланс.

– оновлення статусу замовлення на «виконано» або «відправлено»;

Вхідні дані: новий статус.

Результат: оновлення статусу замовлення в базі даних для подальшого відстеження.

– відображення всього списку товарів, доданих у конкретне замовлення;

Вхідні дані: id замовлення.

Результат: відображення детальної специфікації товарів у кошику клієнта.

– перегляд та видалення зареєстрованих менеджерів (доступне тільки admin);

Вхідні дані для видалення: id moderator.

Результат для видалення: відображення списку менеджерів без обраного (видаленого) співробітника.

– вихід з облікового запису;

Вхідні дані: немає.

Результат: завершення сесії адміністратора/менеджера.

Окрім функцій для співробітників, у системі передбачено публічну частину для клієнтів (покупців). Клієнтам доступні наступні функції:

– реєстрація та авторизація клієнта;

Вхідні дані: електронна пошта, пароль.

Результат: створення особистого облікового запису клієнта та вхід у систему.

– перегляд каталогу та підбір відеокарт;

Вхідні дані: параметри фільтрації (бажаний обсяг пам'яті, ліміт TDP, виробник, ціновий діапазон).

Результат: відображення списку товарів, що відповідають запиту.

– управління кошиком покупок;

Вхідні дані: ідентифікатор товару, кількість.

Результат: додавання товару до кошика, зміна кількості, видалення товару, автоматичний перерахунок загальної вартості.

– оформлення замовлення;

Вхідні дані: контактні дані, адреса доставки, спосіб оплати, перелік товарів у кошику.

Результат: створення нового замовлення в системі (зі статусом «нове» або «очікує обробки») та очищення кошика.

– перегляд історії замовлень (в особистому кабінеті);

Вхідні дані: ідентифікатор авторизованого клієнта.

Результат: відображення списку поточних та попередніх замовлень клієнта з їхніми статусами.

Для формалізації бізнес-логіки та деталізації алгоритмів взаємодії користувачів із вебзастосунком «V-Core», нижче наведено розгорнуті специфікації прецедентів (варіантів використання) для кожної з визначених функціональних вимог системи. Такий підхід дозволяє однозначно трактувати поведінку програмного продукту на різних етапах виконання бізнес-процесів. Кожна специфікація розроблена за стандартизованим шаблоном, що включає визначення дійових осіб, передумов запуску сценарію, основного потоку подій, альтернативних шляхів виконання та постумов. Це гарантує коректну обробку даних та цілісність системи електронної комерції.

Процедуру автоматизованого аналізу технічних параметрів та розрахунку сумісності графічних адаптерів наведено у специфікації прецеденту «Підбір відеокарт» (таблиця А.1).

Таблиця А.1 – Специфікація варіанту використання «Підбір відеокарт»

Характеристика	Опис
Короткий опис	Автоматична перевірка сумісності комплектуючих із системою користувача (блок живлення (БЖ) та корпус) або інтелектуальний підбір за заданими параметрами
Дійові особи	Клієнт
Передумови	Наявність товарів у базі даних із заповненими полями TDP та довжини
Основний потік подій	1. Користувач переходить у розділ «Підбір». 2. Вводить потужність свого БЖ (у ватах) та макс. довжину (у мм). 3. Натискає «Підібрати». 4. Алгоритм відсіює моделі, чий TDP + запас перевищує БЖ, або які довші за вказаний ліміт. 5. Відображається список 100% сумісних карток.
Альтернативний потік подій	2а. Введено некоректні дані (наприклад, від'ємні значення) – клієнтська валідація блокує запит.
Постумови	Користувач отримує технічно обґрунтовані пропозиції

Детальне ознайомлення з асортиментом магазину формалізовано у наступній специфікації (таблиця А.2).

Таблиця А.2 – Специфікація варіанту використання «Перегляд каталогу відеокарт»

Характеристика	Опис
1	**2**
Короткий опис	Користувач переглядає загальний асортимент магазину з фотографіями, цінами та основними характеристиками

Кінець таблиці А.2

1	2
Дійові особи	Клієнт
Передумови	База даних доступна, каталог містить хоча б один товар
Основний потік подій	1. Користувач відкриває головну сторінку або розділ «Каталог». 2. Система відправляє запит до БД на отримання списку товарів. 3. Система відображає сітку карток товарів (пагінація). 4. Користувач може натиснути на товар для перегляду деталей.
Альтернативний потік подій	2а. Помилка з'єднання з БД – система виводить повідомлення: «Не вдалося завантажити товари».
Постумови	Користувач ознайомлений з асортиментом

Процес відбору товарів для майбутньої покупки регламентується таблицею А.3.

Таблиця А.3 – Специфікація варіанту використання «Додавання товару до кошику»

Характеристика	Опис
1	**2**
Короткий опис	Вибір конкретної моделі відеокарти та додавання її до віртуального кошика для подальшої покупки
Дійові особи	Клієнт
Передумови	Користувач знаходиться в каталозі або на сторінці конкретного товару

Кінець таблиці А.3

1	2
Основний потік подій	1. Користувач натискає кнопку «Додати до кошика» біля обраного товару. 2. Система фіксує додавання товару в поточну сесію. 3. Оновлюється індикатор кількості товарів на іконці кошика. 4. Виводиться спливаюче повідомлення про успішне додавання.
Альтернативний потік подій	1а. Товар закінчився на складі – кнопка неактивна або виводиться відповідне повідомлення.
Постумови	Обраний товар знаходиться у віртуальному кошику користувача

Логіку динамічного пошуку та відбору товарів описано у специфікації прецеденту «Фільтрація та сортування відеокарт» (таблиця А.4).

Таблиця А.4 – Специфікація варіанту використання «Фільтрація та сортування відеокарт»

Характеристика	**Опис**
1	**2**
Короткий опис	Динамічний відбір товарів за специфічними технічними параметрами (бренд, пам'ять) та ціною
Дійові особи	Клієнт
Передумови	Користувач перебуває на сторінці каталогу

Кінець таблиці А.4

1	2
Основний потік подій	1. Користувач обирає критерії на панелі фільтрів (наприклад, NVIDIA, 12 ГБ). 2. Обирає тип сортування (наприклад, «Від дешевих до дорогих»). 3. Система формує параметризований запит. 4. Перелік товарів миттєво оновлюється без перезавантаження сторінки.
Альтернативний потік подій	4а. За обраними фільтрами товарів не знайдено – виводиться повідомлення: «Товарів не знайдено. Спробуйте змінити критерії».
Постумови	На екрані відображаються лише релевантні запиту товари

Можливість взаємодії користувачів із платформою через зворотний зв'язок деталізовано в таблиці А.5.

Таблиця А.5 – Специфікація варіанту використання «Публікація коментарів»

Характеристика	**Опис**
1	**2**
Короткий опис	Написання текстового відгуку та виставлення рейтингу конкретному товару
Дійові особи	Клієнт
Передумови	Клієнт авторизований

Кінець таблиці А.5

1	2
Основний потік подій	1. Клієнт відкриває картку товару. 2. Пише текст відгуку та обирає оцінку (від 1 до 5 зірок). 3. Натискає «Опублікувати». 4. Коментар зберігається в БД і з'являється на сторінці товару. 5. Середній рейтинг відеокарти перераховується.
Альтернативний потік подій	2а. Спроба відправити порожній коментар – кнопка блокується системою валідації.
Постумови	Відгук додано до загального списку коментарів товару

Для отримання доступу до розширених функцій системи користувач повинен пройти процес ідентифікації, який регламентується таблицею А.6.

Таблиця А.6 – Специфікація варіанту використання «Авторизація»

Характеристика	**Опис**
1	**2**
Короткий опис	Вхід у систему для отримання доступу до персоналізованих або адміністративних функцій
Дійові особи	Клієнт, Менеджер, Адміністратор
Передумови	Обліковий запис попередньо створено у базі даних

Кінець таблиці А.6

1	2
Основний потік подій	1. Користувач відкриває форму авторизації. 2. Вводить логін (email) та пароль. 3. Система валідує хеш пароля. 4. Система визначає роль та генерує JWT-токен доступу. 5. Відбувається перенаправлення до особистого кабінету або адмін-панелі.
Альтернативний потік подій	3а. Невірний пароль або логін – виводиться повідомлення про помилку.
Постумови	Сесія розпочата, права доступу застосовано

Процес створення нового облікового запису для неавторизованих клієнтів описано в таблиці А.7.

Таблиця А.7 – Специфікація варіанту використання «Реєстрація»

Характеристика	Опис
Короткий опис	Створення нового клієнтського облікового запису
Дійові особи	Клієнт
Передумови	Користувач не авторизований у системі
Основний потік подій	1. Користувач відкриває сторінку реєстрації. 2. Заповнює форму (ПІБ, Email, Пароль). 3. Система перевіряє унікальність Email. 4. Пароль хешується, дані зберігаються у БД. 5. Користувач автоматично авторизується.
Альтернативний потік подій	3а. Email вже існує в БД – система пропонує авторизуватися або відновити пароль.
Постумови	Створено новий профіль клієнта

Функціонал перегляду відібраних до покупки товарів описано в таблиці А.8.

Таблиця А.8 – Специфікація варіанту використання «Перегляд кошику»

Характеристика	Опис
Короткий опис	Відображення переліку доданих до віртуального кошика товарів, їх вартості та загальної суми
Дійові особи	Клієнт
Передумови	До кошика додано хоча б один товар (опціонально)
Основний потік подій	1. Клієнт натискає на іконку кошика в інтерфейсі. 2. Система відображає сторінку або модальне вікно зі списком товарів. 3. Для кожної позиції вказується фото, назва, ціна та обрана кількість. 4. Внизу відображається підсумкова сума до сплати.
Альтернативний потік подій	2а. Кошик порожній – система відображає повідомлення «Ваш кошик порожній» та пропонує перейти до каталогу.
Постумови	Клієнт ознайомлений зі вмістом свого кошика

Редагування кошику описано в таблиці А.9.

Таблиця А.9 – Специфікація варіанту використання «Редагування кошику»

Характеристика	Опис
1	**2**
Короткий опис	Зміна кількості обраних товарів або їх повне видалення з віртуального кошика

Кінець таблиці А.9

1	2
Дійові особи	Клієнт
Передумови	Клієнт перебуває в інтерфейсі перегляду кошика
Основний потік подій	1. Клієнт натискає кнопки «+» або «-» для зміни кількості конкретного товару, або кнопку «Видалити». 2. Система миттєво оновлює дані в сесії. 3. Підсумкова сума замовлення перераховується і оновлюється на екрані.
Альтернативний потік подій	1а. Клієнт зменшує кількість до нуля – товар автоматично видаляється з кошика.
Постумови	Вміст кошика та фінальна вартість актуалізовані

Ключовий бізнес-процес електронної комерції – фіналізація покупки – представлено у специфікації «Оформлення замовлень» (таблиця А.10).

Таблиця А.10 – Специфікація варіанту використання «Оформлення замовлень»

Характеристика	**Опис**
1	**2**
Короткий опис	Процес підтвердження обраних у кошику товарів та фіналізація покупки
Дійові особи	Клієнт
Передумови	Користувач авторизований (або гість), у кошику є хоча б один товар

Кінець таблиці А.10

Основний потік подій	1. Клієнт переходить до оформлення з кошика. 2. Заповнює/підтверджує контакти та адресу доставки. 3. Натискає «Підтвердити замовлення». 4. Система створює запис у таблиці замовлень та віднімає товари зі складу. 5. Клієнт отримує електронний чек або сповіщення.
Альтернативний потік подій	4а. Товар щойно закінчився на складі – транзакція відхиляється з відповідним сповіщенням.
Постумови	Замовлення передано в обробку менеджерам

Таблиця А.11 – Специфікація варіанту використання «Порівняння відеокарт»

Характеристика	**Опис**
1	**2**
Короткий опис	Зіставлення технічних характеристик кількох обраних моделей відеокарт
Дійові особи	Клієнт
Передумови	До списку порівняння додано принаймні два товари
Основний потік подій	1. Клієнт натискає кнопку «Порівняти» біля обраних товарів. 2. Переходить на сторінку порівняння. 3. Система генерує зведену таблицю з детальними характеристиками (обсяг пам'яті, частота, TDP тощо). 4. Відмінності між моделями підсвічуються для зручності.

Кінець таблиці А.11

1	2
Альтернативний потік подій	2а. Додано лише один товар – система пропонує додати ще мінімум одну відеокарту для порівняння.
Постумови	Користувач отримав наочний аналіз обраних моделей

Функціонал керування життєвим циклом оформлених покупок, доступний персоналу магазину, описано в таблиці А.12.

Таблиця А.12 – Специфікація варіанту використання «Обробка замовлень»

Характеристика	Опис
Короткий опис	Керування життєвим циклом замовлень (перегляд, зміна статусів)
Дійові особи	Менеджер, Адміністратор
Передумови	Успішна авторизація з правами Менеджера/Адміна
Основний потік подій	1. Співробітник відкриває розділ «Замовлення» в адмін-панелі. 2. Переглядає деталі нових надходжень. 3. Після зв'язку з клієнтом або відправки змінює статус (напр. «Відправлено»). 4. Зміни зберігаються в БД.
Альтернативний потік подій	3а. Клієнт скасував замовлення – статус змінюється на «Скасовано», товари повертаються на склад.
Постумови	Статус замовлення актуалізовано

Процедуру контролю за користувацьким контентом зі сторони адміністрації наведено в таблиці А.13.

Таблиця А.13 – Специфікація варіанту використання «Модерація коментарів»

Характеристика	Опис
Короткий опис	Контроль за користувацьким контентом, видалення нецензурної лексики або спаму
Дійові особи	Менеджер, Адміністратор
Передумови	Успішна авторизація з відповідними правами
Основний потік подій	1. Співробітник переходить до розділу «Відгуки». 2. Знаходить порушення у тексті коментаря. 3. Натискає кнопку «Видалити». 4. Система видаляє запис із БД та перераховує рейтинг товару.
Альтернативний потік подій	Відсутній (пряма дія).
Постумови	Небажаний контент вилучено з публічної вітрини

Повний цикл управління асортиментом товарів на вітрині регламентується специфікацією «Керування каталогом (CRUD)» (таблиця А.14).

Таблиця А.14 – Специфікація варіанту використання «Керування каталогом (CRUD)»

Характеристика	**Опис**
1	**2**
Короткий опис	Повний цикл управління товарною базою (додавання, зміна характеристик, видалення)
Дійові особи	Адміністратор
Передумови	Успішна авторизація з правами «Admin»

Кінець таблиці А.14

1	2
Основний потік подій	1. Адміністратор відкриває «Управління товарами». 2. Натискає «Додати товар» або редагує існуючий. 3. Заповнює форму (назва, ціна, TDP, пам'ять). 4. Зберігає зміни. 5. Каталог миттєво оновлюється на публічній частині сайту.
Альтернативний потік подій	3а. Введено некоректний тип даних (текст замість ціни) – система не пропускає збереження.
Постумови	База товарів актуалізована

Адміністрування облікових записів співробітників магазину описано в таблиці А.15.

Таблиця А.15 – Специфікація варіанту використання «Управління персоналом»

Характеристика	**Опис**
1	**2**
Короткий опис	Реєстрація нових менеджерів та блокування доступу існуючим співробітникам
Дійові особи	Адміністратор
Передумови	Успішна авторизація з правами «Admin»

Кінець таблиці А.15

1	2
Основний потік подій	1. Адміністратор переходить у розділ «Персонал». 2. Додає нового користувача, призначаючи йому роль «Manager». 3. Система генерує обліковий запис.
Альтернативний потік подій	1а. Адміністратор блокує існуючого менеджера, обмежуючи його доступ до системи.
Постумови	Змінено склад співробітників, які мають доступ до панелі керування

Процес формування звітів та перегляду ключових показників ефективності формалізовано в таблиці А.16.

Таблиця А.16 – Специфікація варіанту використання «Формування статистики та звітності»

Характеристика	Опис
1	**2**
Короткий опис	Формування графіків продажів та визначення популярності товарів
Дійові особи	Адміністратор
Передумови	Наявність успішних замовлень у базі
Основний потік подій	1. Адмін відкриває дашборд. 2. Система агрегує дані про суми замовлень та продані одиниці товару. 3. Відображаються візуальні графіки доходу.
Альтернативний потік подій	2а. За обраний період немає продажів – графік порожній.
Постумови	Адміністратор отримав аналітичну інформацію

Налаштування математичної логіки для модуля підбору комплектуючих деталізовано у специфікації прецеденту «Налаштування параметрів сумісності» (таблиця А.17).

Таблиця А.17 – Специфікація варіанту використання «Налаштування параметрів сумісності»

Характеристика	**Опис**
Короткий опис	Керування технічними правилами для модуля автоматичного підбору відеокарт
Дійові особи	Адміністратор
Передумови	Успішна авторизація з правами «Admin»
Основний потік подій	1. Перехід у налаштування системи. 2. Адмін змінює коефіцієнт запасу потужності БЖ (наприклад, +20% до TDP). 3. Зберігає глобальні налаштування. 4. Алгоритм на клієнтській стороні починає працювати за новими правилами.
Альтернативний потік подій	Відсутній.
Постумови	Логіку розрахунку апаратної сумісності змінено

Таким чином, деталізовані специфікації всіх ключових варіантів використання утворюють повний набір функціональних вимог, що є фундаментальною основою для подальшого етапу проєктування архітектури та розробки бази даних вебзастосунку «V-Core».

3.1.2 Вимоги до організації вхідних та вихідних даних

Усі постійні дані системи (каталог відеокарт, облікові записи користувачів, історія замовлень, відгуки) повинні зберігатися у реляційній базі даних SQLite.

Вхідні дані:

введення даних здійснюється користувачами (клієнтами, менеджерами, адміністратором) через графічний вебінтерфейс за допомогою стандартних пристроїв (клавіатура, миша, сенсорні екрани мобільних пристроїв). Дані можуть:

– вводитися вручну через текстові поля (наприклад, ПІБ, адреса доставки, технічні характеристики відеокарт, текст відгуку);

– обиратися з наявних випадаючих списків, чекбоксів або радіокнопок (наприклад, вибір бренду, типу пам'яті, статусу замовлення, способу оплати);

– задаватися за допомогою інтерактивних елементів (наприклад, повзунки для вибору цінового діапазону у фільтрах підбору відеокарт).

– автоматично завантажуватися з бази даних системи (наприклад, збережені контактні дані авторизованого клієнта, історія попередніх замовлень, актуальні залишки товарів на складі).

Вихідні дані:

виведення даних відбувається на екранах пристроїв користувачів у режимі реального часу після обробки запитів сервером. Формати вихідних даних включають:

– динамічно згенеровані вебсторінки з каталогом товарів та детальними характеристиками відеокарт;

– табличні представлення даних (для зручного перегляду історії замовлень, списку клієнтів або інвентаризації складу в панелі керування).

3.2 Вимоги до надійності

Надійне (стійке) функціонування вебзастосунку має бути забезпечене виконанням комплексу організаційно-технічних заходів, перелік яких наведено нижче:

– резервне копіювання даних: автоматизоване та регулярне створення резервних копій бази даних (включаючи каталог відеокарт, інформацію про клієнтів та історію замовлень) і конфігураційних файлів, а також їх безпечне

зберігання на віддалених серверах для гарантованого відновлення інформації у разі апаратних або програмних збоїв;

– моніторинг та логування: впровадження системи моніторингу, яка цілодобово відстежує стан серверного обладнання та доступність вітрини магазину, а також ведення детальних журналів логування, що реєструють події, транзакції та критичні помилки для швидкого виявлення та усунення проблем адміністраторами;

– безперебійне живлення та резервування: розміщення серверної частини застосунку в надійному дата-центрі (або на професійному хостингу), який забезпечений системами безперебійного живлення (ДБЖ), резервними каналами зв'язку та механізмами автоматичного перемикання на резервні потужності у разі відмови основних вузлів інфраструктури.

3.3 Вимоги до умов експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики вебзастосунку, визначаються санітарними та технічними вимогами, що пред’являються до апаратного забезпечення (серверного обладнання та клієнтських пристроїв) у частині умов їх експлуатації в закритих приміщеннях.

Користувач (клієнт, менеджер або адміністратор) повинен мати базові навички роботи з персональним комп’ютером, ноутбуком або мобільним пристроєм, а також досвід взаємодії із сучасними веббраузерами.

3.4 Вимоги до складу та параметрів технічних засобів

Оскільки система має клієнт-серверну архітектуру, технічні вимоги поділяються на вимоги до робочого місця та вимоги до сервера.

Для повноцінної роботи з панеллю керування (робоче місце менеджера/адміністратора) необхідна система наступної мінімальної конфігурації:

– процесор: двоядерний із тактовою частотою не менше 2,0 ГГц;

– оперативна пам'ять: від 4 ГБ;

– системний диск: 100 ГБ вільного дискового простору (для операційної системи, локального кешу та збереження вивантажених звітів);

– монітор: роздільна здатність не менше 1920x1080 пікселів для комфортної роботи з об'ємними таблицями замовлень та каталогом відеокарт;

– необхідно стабільне мережеве з'єднання з виходом в інтернет.

Для клієнтів (покупців вітрини) мінімальні вимоги зводяться до наявності будь-якого сучасного пристрою (ПК, ноутбук, планшет або смартфон) з підключенням до мережі Інтернет.

Для розгортання серверної частини (мінімальні параметри хостингу або VPS-сервера):

– процесор: від 1 до 2 віртуальних ядер (vCPU);

– оперативна пам'ять: мінімум 2 ГБ (рекомендовано 4 ГБ для стабільної роботи бекенду та бази даних);

– дисковий простір: від 20 ГБ SSD для зберігання БД та медіафайлів (зображень товарів).

3.5 Вимоги до інформаційної та програмної сумісності

– сумісність з операційними системами на клієнтських робочих місцях: операційна система не нижче Windows 10, macOS або сучасні дистрибутиви Linux;

– сумісність з базами даних: наявність системи управління базами даних SQLite (версії 3.x або вище), що використовується для зберігання даних каталогу, користувачів та замовлень;

– сумісність з веббраузерами: оскільки програма має вебінтерфейс, вона повинна бути повністю сумісною та коректно відображатися у сучасних веббраузерах, таких як Google Chrome, Mozilla Firefox, Opera, Safari та Microsoft Edge.

Для розгортання серверної частини вебзастосунку використовується хостинг (або VPS-сервер) із встановленою панеллю керування FastPanel або

аналогічним програмним забезпеченням, що забезпечує зручне адміністрування серверних ресурсів [11]. Основними вимогами до серверного середовища виконання є:

– наявність стабільного інтернет-з’єднання, оскільки вся взаємодія із системою (обробка замовлень, оновлення каталогу) відбувається онлайн;

– сервер із параметрами, достатніми для коректної роботи вебдодатку (рекомендовано принаймні 1-2 віртуальних ядра процесора (vCPU), 2-4 ГБ оперативної пам’яті та від 20 ГБ вільного дискового простору SSD);

– операційна система на сервері, сумісна з обраними технологіями (наприклад, Linux-дистрибутиви Ubuntu або Debian);

– наявність підтримуваної версії вебсервера (Nginx або Apache), що забезпечує обробку HTTP-запитів, роздачу статики та маршрутизацію (Reverse Proxy) до бекенд-додатку;

– підтримка необхідного середовища виконання для бекенд-частини (середовище Node.js версії 18 або вище для забезпечення роботи REST API та взаємодії з СУБД SQLite).

Крім того, для повноцінної роботи з вебзастосунком на стороні користувача (клієнта або менеджера) потрібно мати пристрій (ПК, ноутбук або смартфон) із сучасним веббраузером, що має актуальні оновлення, та доступ до мережі Інтернет.

3.6 Вимоги до маркування та пакування

Продукт необхідно підготувати для розгортання (деплою) на хостингу з панеллю FastPanel або на виділеному Linux-сервері. Для цього проєкт повинен бути зібраний (build) у готові для виконання артефакти (наприклад, архів вихідного коду серверної частини із залежностями package.json та збудовані статичні файли для клієнтської частини) і завантажений на сервер.

У комплект поставки програмного продукту (архів з релізом) повинні входити:

– Файл із описом ReadMe.txt (або README.md), який містить детальні інструкції з розгортання, налаштування змінних середовища (доступи до БД, порти) та використання програми.

– Файл-дамп vcore_db.sql, у якому наведено початкову структуру бази даних для швидкого імпорту на сервері баз даних SQLite.

4 Вимоги до програмної документації

Склад програмної документації, що розробляється та супроводжує вебзастосунок, повинен включати в себе:

– технічне завдання;

– програму та методику випробувань;

– керівництво користувача (з інструкціями для клієнтів, менеджерів та адміністратора);

– опис програми (опис архітектури, структури бази даних та логіки роботи);

– вихідний код програми.

5 Техніко-економічні показники

У даній кваліфікаційній роботі техніко-економічні показники не розраховуються.

6 Етапи роботи

Етапи роботи представлені у таблиці А.18.

Таблиця А.18 – Етапи роботи

Номер	Назва етапів роботи	Термін виконання
1.	Аналіз практичної задачі інженерії ПЗ	17.02.2026
2.	Аналіз вимог до ПЗ	28.03.2026
3.	Розробка архітектури ПЗ	08.04.2026
4.	Розробка проєкту ПЗ	15.04.2026
5.	Реалізація та тестування ПЗ	28.04.2026

7 Порядок контролю та приймання

На кожному етапі створення програмного забезпечення контролюють процеси аналізу та проєктування його складових з урахуванням вимог, викладених у технічному завданні. Кожна фаза розробки має бути представлена у визначені терміни та погоджена із замовником.

Приймання виконується за затвердженою методикою і програмою випробувань, а результати фіксують у протоколі проведення перевірок. Якщо під час приймання програмного продукту виявлено помилки, складають відповідний акт, у якому їх описують. Цей документ підписують представники замовника й розробника, а затверджують керівники обох сторін. Розробник зобов’язаний упродовж не більше двох тижнів виправити всі виявлені недоліки та повідомити замовника про готовність до повторної перевірки.

ДОДАТОК Б – Опис програмного забезпечення «V-Core»

1 Загальні відомості

Позначення та найменування програми: Вебзастосунок «V-Core».

Повна назва: Вебзастосунок для автоматизації обробки онлайн-замовлень та підбору відеокарт для магазину «V-Core».

Програмне забезпечення, необхідне для функціонування програми:

Серверна частина (Хостинг / VPS):

– Операційна система: Linux (рекомендовано дистрибутиви Ubuntu або Debian), macOS або Windows, що підтримує середовище Node.js та СУБД SQLite.

– Середовище виконання: Node.js (версії 18.x або новішої).

– Система управління базами даних (СУБД): SQLite версії 8.0 або новішої.

– Вебсервер: Nginx або Apache (використовується для роздачі скомпільованих статичних файлів клієнтської частини та як reverse proxy для бекенд-додатка).

– Менеджер пакетів: npm або pnpm.

Клієнтська частина (Пристрій користувача):

– Сучасний веббраузер з підтримкою HTML5, CSS3 та стандарту ECMAScript 6+ (наприклад, Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge останніх версій). Додаток є кросплатформним та підтримує як десктопні, так і мобільні версії браузерів.

Мови програмування та базові технології, на яких написана програма:

– Клієнтська частина (фронтенд): TypeScript, JavaScript, HTML5, CSS3 (з використанням бібліотеки React 18, збирача Vite та CSS-фреймворку Tailwind CSS).

– Серверна частина (бекенд): JavaScript / TypeScript (на базі середовища виконання Node.js для побудови REST API).

– Робота з базами даних: Мова структурованих запитів SQL.

2 Функціональне призначення

Призначення програми: Вебзастосунок «V-Core» призначений для автоматизації процесів роздрібної торгівлі відеокартами та надання користувачам інструментарію для підбору комплектуючих з урахуванням апаратної сумісності. Програма забезпечує:

− Інтерактивний перегляд каталогу: Відображення актуального асортименту відеокарт NVIDIA та AMD з детальними технічними характеристиками (архітектура, частоти, типи пам’яті, габарити).

− Багатофакторну фільтрацію: Можливість швидкого відбору товарів за брендом, обсягом відеопам’яті, ціновим діапазоном та рівнем енергоспоживання (TDP).

Підбір за параметрами системи:

− Автоматична перевірка сумісності відеокарти з блоком живлення користувача (розрахунок необхідного запасу потужності).

− Перевірка фізичної сумісності за довжиною GPU та габаритами корпусу.

− Порівняння характеристик: Можливість зіставлення декількох моделей відеокарт у єдиній таблиці для аналізу переваг кожної архітектури.

− Управління замовленнями (Кошик): Формування списку покупок, автоматичний перерахунок вартості та збереження контактних даних для доставки.

− Систему зворотного зв’язку: Можливість авторизованих користувачів залишати відгуки та оцінювати товари, що впливає на загальний рейтинг продукції.

− Адміністрування контенту: Захищений інтерфейс для додавання/редагування товарів, модерації коментарів та моніторингу наявності продукції на складі.

Відомості про функціональні обмеження при використанні:

– Залежність від мережі: Функціонування системи (оновлення залишків, авторизація) потребує стабільного HTTP-з’єднання з серверною частиною (Node.js/SQLite).

– Область спеціалізації: Алгоритми сумісності налаштовані виключно для відеокарт та не охоплюють перевірку сумісності інших компонентів (процесорів, материнських плат тощо).

– Апаратні ресурси: Швидкість обробки фільтрів та складних запитів у каталозі залежить від продуктивності клієнтського пристрою (через використання рендерингу на стороні клієнта – React) та потужності серверної БД.

– Дані: Система не передбачає автоматичного парсингу цін із зовнішніх джерел; актуалізація асортименту покладається на адміністратора через відповідну панель.

3 Опис логічної структури

Структура програми з описом функцій складових частин і зв’язку між ними: Програмне забезпечення вебсистеми магазину «V-Core» має клієнт-серверну архітектуру. Логічно застосунок поділяється на серверну (Backend) та клієнтську (Frontend) частини, кожна з яких складається з окремих функціональних блоків (підсистем) та файлів-модулів:

Точка входу та кореневий компонент (main.tsx, App.tsx): Виконують роль ініціалізаторів клієнтського застосунку. main.tsx є головним файлом-модулем, що монтує React-додаток у DOM-дерево браузера. App.tsx (кореневий компонент) налаштовує клієнт для кешування запитів (React Query), підключає глобальні провайдери контексту (Auth, Cart, GPU, Compare) та ініціалізує маршрутизатор (React Router) для навігації між сторінками.

Підсистема маршрутизованих сторінок (директорія pages/): Кожна сторінка є окремим логічним модулем, що об'єднує UI-компоненти та бізнес-логіку для конкретного маршруту:

– CatalogPage.tsx: Відображає повний каталог відеокарт з підключеною панеллю багаторівневих фільтрів.

– SelectorPage.tsx: Обробляє логіку введення параметрів системи користувача (потужність блоку живлення, розміри корпусу) та ініціює запит до сервісу перевірки сумісності.

– CartPage.tsx: Керує списком обраних товарів та процесом оформлення замовлення (Checkout), включаючи валідацію контактних даних перед відправкою на сервер.

– AdminPage.tsx: Надає закритий інтерфейс для управління базою товарів (CRUD-операції) та модерації користувацьких відгуків.

Підсистема інтерфейсу користувача (директорія components/): Містить незалежні, перевикористовувані UI-компоненти:

– GpuCard.tsx: Відповідає за візуальне представлення конкретної моделі відеокарти, її ключових характеристик, ціни та кнопок взаємодії (додати до кошика, порівняти).

– GpuFilters.tsx: Керує станом фільтрів каталогу (бренд, обсяг пам'яті, ціновий діапазон) та передає параметри для оновлення списку товарів.

– Директорія ui/: Містить базові компоненти (кнопки, поля введення, модальні вікна, сповіщення), побудовані на базі бібліотеки Shadcn UI.

Підсистема управління станом (директорія context/):

– GpuContext.tsx: Зберігає завантажений з сервера каталог товарів та надає методи для його оновлення.

– CartContext.tsx: Зберігає обрані до кошика товари та автоматично розраховує загальну вартість замовлення.

– AuthContext.tsx: Зберігає токен або сесію користувача, визначаючи його рівень доступу та роль (Клієнт або Адміністратор).

Серверна частина (Node.js + Express REST API): Обробляє HTTP-запити від клієнтського додатку, взаємодіє з базою даних SQLite та реалізує ключову бізнес-логіку через відповідні програмні модулі (сервіси та контролери):

– CatalogService: Забезпечує вибірку товарів з бази даних, пошук та серверну фільтрацію.

– CompatibilityService: Реалізує алгоритми сумісності, зіставляючи введені користувачем параметри ПК з технічними характеристиками відеокарт (TDP, довжина).

– OrderService: Відповідає за прийом, валідацію та збереження нових замовлень у базу даних.

Алгоритм програми (загальний): Робота вебзастосунку «V-Core» відбувається за наступним сценарієм:

– Користувач взаємодіє з клієнтським вебзастосунком (Single Page Application), завантаженим у браузер.

– На сторінці підбору (SelectorPage) або в каталозі користувач вводить параметри своєї системи (потужність блоку живлення у Вт, максимальна довжина GPU у мм) або обирає фільтри (виробник, обсяг пам'яті).

– Бібліотека управління формами (React Hook Form) збирає введені дані, а схема валідації (Zod) миттєво перевіряє їх на коректність на стороні клієнта (наприклад, відсутність від'ємних значень або недопустимих символів).

– У разі успішної валідації клієнтський застосунок за допомогою інструменту управління асинхронним станом (TanStack React Query) формує HTTP-запит до бекенду (REST API на базі Node.js).

– Серверна частина отримує параметри запиту та виконує звернення до бази даних SQLite для отримання базового переліку відеокарт.

– На сервері спрацьовує бізнес-логіка (CompatibilityService): алгоритм розраховує необхідний запас потужності та відсіює моделі, які перевищують ліміти споживання або не помістяться у фізичні габарити корпусу клієнта.

– Результати підбору (масив об'єктів відеокарт з їхніми характеристиками та цінами) повертаються на клієнтську частину у форматі JSON.

– React оновлює віртуальний DOM (Virtual DOM) і динамічно відображає відфільтровані картки товарів без перезавантаження сторінки.

– При натисканні кнопки «Додати в кошик» обраний товар зберігається у глобальному стані клієнта (CartContext), а під час оформлення замовлення (Checkout) – відправляється POST-запитом на сервер для фіксації в базі даних та резервування відповідної кількості на складі.

Методи, що використовуються:

– Методи побудови інтерфейсу: Використання компонентного підходу (React) для створення незалежних та перевикористовуваних UI-елементів. Рендеринг інтерфейсу відбувається на стороні клієнта (CSR – Client-Side Rendering).

– Методи стилізації: Застосування утилітарного підходу (Tailwind CSS) для формування адаптивного дизайну безпосередньо в розмітці компонентів без створення окремих файлів стилів.

– Методи управління станом: Використання React Context API для передачі глобальних UI-даних (авторизація, кошик, порівняння) між компонентами без проблеми "prop drilling". Для управління станом серверних даних (кешування, фонове оновлення) застосовується TanStack React Query.

– Методи обробки форм та валідації: Застосування бібліотеки React Hook Form для ефективного збору даних та декларативний опис схем (Zod) для суворої перевірки введених користувачем параметрів перед їх відправкою на сервер.

– Методи взаємодії з сервером: Виконання асинхронних HTTP-запитів (GET, POST, PUT, DELETE) за допомогою REST API формату обміну даними JSON.

– Методи обробки даних на сервері: Використання контролерів для маршрутизації API-запитів, сервісів для реалізації бізнес-логіки та параметризованих SQL-запитів для безпечної взаємодії з реляційною базою даних.

Зв’язки між компонентами:

– Користувач взаємодіє з візуальними компонентами на маршрутизованих сторінках у директорії pages/.

– Компоненти сторінок використовують локальні UI-модулі з components/ та звертаються до глобального стану або ініціюють дії через хуки у context/.

– Інструменти управління станом формують HTTP-запити та передають їх до серверного REST API.

– Серверне API обробляє запити, звертається до БД SQLite, виконує необхідну логіку та повертає відповідь клієнту, після чого React автоматично оновлює відповідні частини інтерфейсу.

Зв’язок програми з іншими програмами:

– СУБД SQLite: Вебзастосунок взаємодіє з локальною реляційною базою даних для довгострокового зберігання каталогу відеокарт, облікових записів адміністраторів та історії замовлень.

– Вебсервер (Nginx / Node.js): Середовище Node.js виконує роль сервера застосунків (Application Server), обробляючи логіку API. Nginx може використовуватися як вебсервер для хостингу статичних файлів клієнтського застосунку (HTML, JS, CSS, зображення) та як зворотний проксі (reverse proxy) для перенаправлення запитів до Node.js.

– Веббраузер користувача: Виступає середовищем виконання клієнтської частини програми (використовує рушій V8 або аналоги для виконання JavaScript, здійснює рендеринг DOM), відповідає за безпосередню взаємодію з користувачем та візуалізацію інтерфейсу.

4 Необхідні технічні засоби

Для роботи вебзастосунку «V-Core» необхідні наступні технічні засоби:

Серверна частина (Хостинг / VPS):

Мінімальні вимоги (для базового функціонування та невеликої кількості товарів):

– Процесор (vCPU): 2-ядерний, з тактовою частотою 2.0 ГГц.

– Оперативна пам’ять (RAM): 4 ГБ.

– Дисковий простір: 20 ГБ вільного місця (для операційної системи, середовища Node.js, СУБД SQLite та базових зображень товарів).

Рекомендовані вимоги (для швидкої обробки запитів при великій кількості одночасних користувачів та розширеному каталозі):

– Процесор (vCPU): 4 або більше ядер, з тактовою частотою 2.5 ГГц або вище.

– Оперативна пам’ять (RAM): 8 ГБ або більше.

– Дисковий простір: 50 ГБ або більше вільного місця на SSD-накопичувачі (для швидкого доступу бази даних до кешу та зберігання високоякісних медіафайлів вітрини).

Клієнтська частина (Пристрій користувача):

– Будь-який персональний комп’ютер, ноутбук, планшет або смартфон, здатний запустити та комфортно відображати сучасний веббраузер (завдяки адаптивному дизайну на базі Tailwind CSS застосунок коректно працює на різних діагоналях екранів).

– Підключення до мережі Інтернет для доступу до сервера, на якому розгорнуто вебзастосунок «V-Core».

– Пристрої вводу-виводу: клавіатура, маніпулятор типу «миша» (або сенсорний екран пристрою), монітор.

5 Виклик і завантаження

Спосіб виклику програми з відповідного носія даних:

– Серверна частина (Бекенд REST API): Програмний код серверної частини розміщується на серверному комп’ютері (або VPS). Запуск серверного додатка здійснюється з командного рядка у середовищі Node.js. Для стабільної роботи в продакшені (production) зазвичай використовується менеджер процесів (наприклад, PM2).

Команда для запуску може виглядати так:

- pm2 start server.js --name "vcore-api" Для локального тестування та розробки використовується команда з автоматичним перезавантаженням:

- npm run dev

– Клієнтська частина (Фронтенд SPA): Файли клієнтського застосунку компілюються інструментом Vite (npm run build) і розміщуються на вебсервері (Nginx або Apache) для статичної роздачі. Доступ до вебзастосунку «V-Core» здійснюється користувачем через будь-який сучасний веббраузер. Для цього необхідно ввести в адресному рядку браузера URL-адресу магазину (доменне ім'я, наприклад, https://vcore-shop.com, або http://localhost:5173 при локальному запуску).

Вхідні точки в програму: Оскільки застосунок побудовано за архітектурою Single Page Application (SPA), вхідні точки розділяються на клієнтську маршрутизацію (за яку відповідає React Router) та серверні точки доступу (REST API ендпоінти):

Клієнтські маршрути (інтерфейс користувача):

– / (Home): Головна (landing) сторінка магазину.

– /catalog: Сторінка загального каталогу відеокарт із панеллю фільтрації.

– /selector: Сторінка підбору комплектуючих за параметрами системи (TDP, габарити).

– /cart: Кошик покупок та перехід до оформлення замовлення.

– /admin: Захищена панель адміністратора.

Основні серверні HTTP-ендпоінти (REST API):

– GET /api/gpus: Ендпоінт для отримання каталогу відеокарт. Приймає параметри запиту (query params) для фільтрації за брендом, ціною, пам'яттю тощо. Повертає масив об'єктів у форматі JSON.

– GET /api/gpus/{id}: Отримання детальної інформації про конкретну модель відеокарти.

– POST /api/orders: Ендпоінт для створення нового замовлення. Приймає JSON із даними кошика та контактною інформацією клієнта.

– POST /api/auth/login: Ендпоінт авторизації користувачів та адміністраторів. Повертає JWT-токен для доступу до захищених маршрутів.

6 Вхідні дані

Характер, організація і попередня підготовка вхідних даних:

– Параметри системи користувача (для модуля підбору): Числові значення, що описують технічні характеристики комп'ютера клієнта. Вводяться користувачем безпосередньо у відповідні поля або за допомогою повзунків (слайдерів) на сторінці підбору. Основні параметри: потужність наявного блоку живлення (у ватах) та максимальна допустима довжина відеокарти (у міліметрах). Попередньої підготовки не потребують.

– Параметри фільтрації каталогу: Дані для сортування та відбору товарів. Обираються користувачем з випадаючих списків або чекбоксів (виробник GPU, тип та обсяг пам'яті, серія, ціновий діапазон).

– Дані авторизації та реєстрації: Текстові рядки (ім'я, електронна пошта, пароль), які користувач вводить у форми доступу. Перед відправкою проходять автоматичну попередню перевірку (валідацію) на стороні клієнта за допомогою бібліотеки Zod.

– Дані для оформлення замовлення: Текстова інформація (адреса доставки, номер телефону, ПІБ отримувача) та масив даних з кошика покупок (ідентифікатори товарів, кількість).

– Контент каталогу (адміністративні дані): Технічні специфікації нових відеокарт (назва, архітектура, частоти, TDP, ціна, залишок на складі) та зображення товарів. Вводяться адміністратором через захищений інтерфейс. Зображення можуть потребувати попередньої підготовки (масштабування та оптимізації розміру файлу).

Формат, описання та спосіб кодування вхідних даних:

– Загальний формат передачі між клієнтом та сервером: Усі структуровані вхідні дані (форми, замовлення) передаються тілом HTTP-запиту (Body) у форматі JSON (JavaScript Object Notation). Кодування тексту – UTF-8.

– Параметри пошуку та фільтрації: Передаються як параметри URL-запиту (Query Parameters) для забезпечення можливості ділитися посиланням на

результати (наприклад, ?brand=NVIDIA&minPrice=15000&minTdp=200). Кодування – URL-encoded.

– Медіадані (зображення товарів): Передаються на сервер у бінарному вигляді (через FormData) у форматах .jpg, .png або .webp.

Опис вхідних даних у структурі бази даних SQLite (сутності):

– Користувачі (User): email (VARCHAR, унікальний), password_hash (VARCHAR, хешований рядок), role (ENUM: 'user', 'admin').

– Параметри сумісності (GPU): tdp_watts (INTEGER, значення у ватах), length_mm (INTEGER, значення у міліметрах), interface (VARCHAR).

– Фінансові дані (Order / GPU): price, total_amount (DECIMAL(10,2), числа з плаваючою крапкою для фінансових розрахунків).

– Відгуки (Comment): text (TEXT), rating (INTEGER, значення від 1 до 5). Кодування бази даних – UTF-8 (utf8mb4 для повної підтримки спеціальних символів).

7 Вихідні дані

Характер і організація вихідних даних:

− Результати роботи каталогу та модуля підбору: Список відеокарт, що відповідають обраним користувачем фільтрам або апаратним параметрам системи (з урахуванням перевірки на сумісність за TDP та габаритами).

− Результати оформлення замовлення: Сторінка підтвердження (електронний чек) із зазначенням унікального номера замовлення (Order ID), переліком обраних товарів, фіксованою підсумковою вартістю та поточним статусом обробки.

− Результати обробки відгуків та рейтингу: Оновлений масив коментарів до конкретної моделі відеокарти та перерахований середній рейтинг товару.

− Дані сесії авторизації: Цифровий токен доступу (JWT) та об'єкт з базовими даними профілю (роль користувача, ім'я) для розмежування доступу до клієнтських маршрутів.

− Інформаційні повідомлення для користувача: Системні сповіщення про статус виконання операцій (наприклад, «Товар успішно додано до кошика», «Замовлення прийнято в обробку», «Помилка валідації даних»).

Формат, опис і спосіб кодування вихідних даних:

– Усі відповіді від серверного REST API (Node.js) передаються на клієнтську частину виключно у форматі JSON. Кодування всього текстового контенту – UTF-8.

Відповідь API каталогу (наприклад, GET /api/gpus):

− Масив JSON-об'єктів. Базова структура об'єкта товару виглядає наступним чином:

− { "id": "string", "name": "string", "brand": "string", "price": "number", "tdpWatts": "number", "inStock": "boolean", "imageUrl": "string" }

Відповідь API створення замовлення (наприклад, POST /api/orders):

− JSON-об'єкт підтвердження операції:

− { "orderId": "string", "status": "string", "totalAmount": "number", "message": "string" }

Відображення на клієнтській стороні (у браузері користувача):

– Каталог та результати підбору: Отримані JSON-дані динамічно обробляються бібліотекою React та рендеряться у вигляді HTML-розмітки за допомогою перевикористовуваних компонентів (наприклад, GpuCard.tsx).

– Інтерфейс кошика: Динамічна HTML-таблиця, яка миттєво перераховує підсумкові суми за допомогою React Context (CartContext) при зміні кількості товарів.

– Інформаційні повідомлення: Відображаються у вигляді сучасних спливаючих toast-повідомлень (за допомогою бібліотеки Sonner), що з'являються поверх основного інтерфейсу та автоматично зникають через заданий проміжок часу.

– Все візуальне та текстове відображення на стороні клієнта використовує кодування UTF-8, що забезпечує коректну роботу з українською мовою.

ДОДАТОК В – Програма та методика випробувань вебзастосунку «V-Core»

1 Об’єкт випробувань

Найменування випробуваної програми: Вебзастосунок для автоматизації обробки онлайн-замовлень та підбору відеокарт для магазину «V-Core».

Область застосування випробуваної програми:

– Електронна комерція та роздрібний продаж комп'ютерних комплектуючих (відеокарт).

– Підбір обладнання з автоматичною перевіркою апаратної сумісності (за рівнем енергоспоживання TDP та фізичними габаритами).

– Оформлення, обробка та супровід клієнтських замовлень в онлайн-режимі.

– Адміністрування бази товарів каталогу та управління користувацьким контентом (модерація відгуків і рейтингів).

Позначення випробуваної програми: Вебзастосунок «V-Core».

2 Мета випробувань

Метою проведення випробувань вебзастосунку «V-Core» є:

– Перевірка відповідності вимогам: Підтвердження того, що розроблений вебзастосунок повністю відповідає функціональним та нефункціональним вимогам, визначеним у Технічному завданні.

– Валідація алгоритму сумісності: Перевірка коректності роботи інтелектуального модуля підбору відеокарт, зокрема точності розрахунків енергоспоживання (TDP) та фізичної сумісності за габаритами.

– Контроль функціональності каталогу: Підтвердження стабільної роботи систем багатофакторної фільтрації та сортування товарів за різними технічними параметрами.

– Перевірка транзакційної цілісності: Валідація повного циклу оформлення замовлення: від додавання товару в кошик до коректного збереження даних у базі даних SQLite та оновлення складських залишків.

– Аудит безпеки та доступу: Перевірка надійності системи авторизації на базі JWT-токенів та коректності розмежування прав доступу між звичайними користувачами та адміністраторами.

– Оцінка інтерфейсу та UX: Забезпечення того, що користувацький інтерфейс, побудований на React та Tailwind CSS, є інтуїтивно зрозумілим, чуйним (responsive) та коректно відображається на різних типах пристроїв.

– Стрес-тестування обробки даних: Перевірка стійкості вебзастосунку до введення некоректних даних (валідація через Zod) та стабільності роботи при помилках мережевої взаємодії з REST API.

– Виявлення дефектів: Своєчасне знаходження та документування помилок у логіці, дизайні або роботі бази даних для їх подальшого усунення перед введенням системи в експлуатацію.

3 Вимоги до програми

Під час випробувань перевірці підлягають наступні вимоги до вебзастосунку «V-Core», задані в Технічному завданні:

Функціональні вимоги:

– Коректне відображення вітрини (каталогу) відеокарт з можливістю багатофакторної фільтрації товарів за брендом, обсягом відеопам'яті, ціновим діапазоном та іншими технічними характеристиками.

– Безпомилкова робота модуля підбору: коректне відсіювання відеокарт, що не проходять перевірку апаратної сумісності на основі введених користувачем параметрів (потужності наявного блоку живлення у Вт та максимальної допустимої довжини у мм).

– Можливість реєстрації, безпечної авторизації та виходу з облікового запису (управління сесіями на базі JWT-токенів).

– Функціонування кошика покупок: додавання товару, видалення, зміна кількості позицій та миттєвий автоматичний перерахунок загальної вартості замовлення.

– Успішне проходження флоу оформлення замовлення (Checkout) із валідацією контактних даних клієнта, фіксацією замовлення у базі даних та оновленням залишків на складі.

– Можливість додавання відеокарт до списку порівняння для наочного зіставлення їхніх характеристик.

– Можливість для авторизованих користувачів залишати текстові відгуки та виставляти рейтингові оцінки товарам.

– Коректна робота захищеної панелі адміністратора: управління контентом (CRUD-операції для відеокарт: створення, читання, оновлення, видалення) та модерація залишених коментарів.

Вимоги до надійності:

– Коректна та своєчасна валідація вхідних даних від користувача як на стороні клієнта (за допомогою Zod), так і на стороні сервера (захист від введення некоректних типів даних, від'ємних значень параметрів системи або невалідних email-адрес).

– Стійкість клієнтського застосунку (SPA) до помилок мережевої взаємодії (наприклад, при тимчасовій недоступності REST API або помилках бази даних). Збій запиту не повинен призводити до краху (білого екрана) всього інтерфейсу.

– Забезпечення узгодженості даних: недопущення оформлення замовлення на товар, який вже закінчився на складі (out of stock).

– Обов'язкове інформування користувача про успішне виконання операцій або виникнення помилок за допомогою зрозумілих текстових сповіщень (toast-повідомлень).

Вимоги до інтерфейсу користувача:

– Сучасний, інтуїтивно зрозумілий та адаптивний (responsive) дизайн, що забезпечує коректне відображення та зручність використання як на десктопних моніторах, так і на екранах мобільних пристроїв.

– Чітке візуальне виділення ключових елементів взаємодії (кнопки «Додати в кошик», індикатори сумісності/несумісності, ціни товарів).

– Відсутність затримок (зависань) інтерфейсу під час перемикання між сторінками завдяки використанню клієнтської маршрутизації (React Router).

4 Вимоги до програмної документації

Склад програмної документації, що пред'являється на випробування:

– Технічне завдання на розробку вебзастосунку «V-Core».

– Опис програмного забезпечення вебзастосунку «V-Core».

– Програма та методика випробувань вебзастосунку «V-Core».

– Керівництво користувача (оператора) вебзастосунку «V-Core».

– Текст програми (лістинги вихідного коду ключових модулів).

Спеціальні вимоги: Спеціальні вимоги до програмної документації, окрім її повноти, достовірності та відповідності загальноприйнятим стандартам оформлення інженерної документації, у Технічному завданні не вказані.

5 Засоби і порядок випробувань

Технічні засоби, що використовуються під час випробувань:

Серверна частина:

– Процесор: Intel Core i5 (або аналогічний) 4+ ядер, 2.5+ ГГц.

– Оперативна пам’ять: 8 ГБ DDR4.

– Накопичувач: SSD 256 ГБ (з достатнім вільним місцем для ОС, середовища виконання, бази даних та медіафайлів).

– Операційна система: Linux (наприклад, Ubuntu 22.04 LTS) або Windows 10/11.

Клієнтська частина:

– Персональний комп’ютер, ноутбук або мобільний пристрій з підключенням до мережі Інтернет.

– Сучасний веббраузер: Google Chrome (остання версія), Mozilla Firefox або Safari.

Програмні засоби, що використовуються під час випробувань:

Серверна частина:

– Середовище виконання Node.js (версії 18+).

– СУБД SQLite версії 8.0+.

– Менеджер пакетів npm або pnpm.

Клієнтська частина:

– Веббраузер з інструментами розробника (DevTools) та розширенням React Developer Tools для інспектування стану компонентів і мережевих запитів.

Інструментальні засоби:

– Середовище розробки: WebStorm від JetBrains (для перегляду коду, логів та відлагодження).

– Система контролю версій: Git.

– Фреймворки для тестування: Vitest та React Testing Library (для перевірки UI-компонентів).

– Інструменти збірки: Vite.

Порядок проведення випробувань:

Підготовчий етап:

– Розгортання вебзастосунку «V-Core» у тестовому середовищі: клонування репозиторію та встановлення залежностей (npm install).

– Налаштування та запуск сервера СУБД SQLite.

– Створення структури реляційної бази даних (таблиць User, GPU, Order, Comment) та імпорт тестового набору даних (seeding), що містить репрезентативний каталог відеокарт різних поколінь (від офісних до флагманських рішень), а також тестові облікові записи адміністратора та клієнтів.

– Запуск серверної та клієнтської частини у режимі розробки (npm run dev).

– Перевірка доступності веб-інтерфейсу вебзастосунку «V-Core» через браузер за локальною адресою (наприклад, http://localhost:5173).

Основний етап (виконання тестів):

– Проведення функціонального тестування основного бізнес-процесу: відвідування каталогу, використання фільтрів, перевірка роботи модуля підбору (введення різних значень потужності БЖ та довжини корпусу).

– Проведення наскрізного (End-to-End) тестування процесу купівлі: реєстрація, додавання сумісного товару в кошик, оформлення замовлення (Checkout).

– Проведення тестування адміністративної панелі (додавання нових товарів, модерація відгуків).

– Тестування обробки помилок та стійкості: перевірка валідації форм (через бібліотеку Zod) та реакції інтерфейсу на некоректні дані.

– Фіксація результатів кожного тест-кейсу: опис кроків, очікуваний результат, фактичний результат, статус (пройдено/не пройдено).

Завершальний етап:

– Аналіз отриманих результатів тестування.

– Порівняння фактичних результатів з очікуваними вимогами технічного завдання.

– Виявлення та документування всіх невідповідностей, UI-дефектів (наприклад, некоректного відображення на мобільних пристроях) та логічних помилок.

– Складання фінального звіту про результати випробувань.

6 Методи випробувань

Випробування вебзастосунку «V-Core» проводяться з використанням наступних методів:

Метод функціонального тестування (ручне наскрізне тестування): Цей метод передбачає перевірку відповідності реалізованих функцій ПЗ вимогам, зазначеним у Технічному завданні, шляхом безпосередньої взаємодії з користувацьким інтерфейсом та імітації реальних сценаріїв використання (User Flows).

Тест-кейс 1: Перевірка базової функціональності каталогу та фільтрів.

– Опис: Перевірка коректності відображення асортименту та роботи багатофакторної фільтрації.

– Кроки:

1) Відкрити сторінку каталогу вебзастосунку.

2) Обрати фільтр за брендом (наприклад, «NVIDIA»).

3) Задати ціновий діапазон (наприклад, від 15000 до 30000 грн).

4) Обрати мінімальний обсяг відеопам'яті (наприклад, 12 ГБ).

5) Перевірити сформований список товарів.

6) Скинути фільтри кнопкою «Очистити».

– Очікуваний результат: Система миттєво (завдяки React) оновлює список карток товарів. Усі відображені відеокарти строго відповідають заданим критеріям. При скиданні фільтрів відображається повний каталог.

Тест-кейс 2: Перевірка модуля підбору (сумісності).

– Опис: Перевірка алгоритму відсіювання апаратно несумісних комплектуючих.

– Кроки:

1) Перейти на сторінку «Підбір відеокарти».

2) Ввести у поле «Потужність блоку живлення» значення 500 Вт.

3) Ввести у поле «Максимальна довжина» значення 250 мм.

4) Натиснути «Підібрати».

5) Змінити потужність на 850 Вт та довжину на 350 мм, повторити підбір.

– Очікуваний результат: У першому випадку флагманські відеокарти (наприклад, RTX 4090 з TDP 450W та довжиною 340 мм) не відображаються у

видачі, оскільки алгоритм розраховує нестачу живлення та фізичну неможливість встановлення. У другому випадку флагмани з'являються у списку.

Тест-кейс 3: Перевірка функціональності кошика та управління станом.

– Опис: Перевірка логіки CartContext та автоматичних фінансових розрахунків.

– Кроки:

1) Додати товар «А» (ціна 10000) до кошика.

2) Додати товар «Б» (ціна 15000) до кошика.

3) Відкрити кошик, перевірити загальну суму.

4) Збільшити кількість товару «А» до 2 одиниць.

5) Видалити товар «Б» з кошика.

– Очікуваний результат: Загальна сума коректно перераховується на кожному кроці (Крок 3: 25000; Крок 4: 35000; Крок 5: 20000). Індикатор (badge) на іконці кошика в шапці сайту синхронно оновлюється.

Тест-кейс 4: Перевірка оформлення замовлення (Checkout) та роботи з БД.

– Опис: Валідація процесу створення замовлення та збереження даних.

– Кроки:

1) Маючи товари в кошику, натиснути «Оформити замовлення».

2) Заповнити форму (ПІБ, телефон, адреса) коректними даними.

3) Натиснути «Підтвердити покупку».

4) Авторизуватися як адміністратор та перевірити панель замовлень.

– Очікуваний результат: Клієнт бачить сторінку успішного оформлення з номером замовлення (кошик при цьому очищується). В адміністративній панелі з'являється новий запис із правильними даними клієнта та прив'язаними товарами. Залишок товарів на складі в БД зменшується.

Тест-кейс 5: Перевірка рольової моделі (RBAC) та панелі адміністратора.

– Опис: Перевірка доступу до закритих маршрутів та CRUD-операцій.

– Кроки:

1) Спробувати зайти на сторінку /admin у статусі гостя.

2) Авторизуватися під обліковим записом клієнта (role: user) та спробувати зайти на /admin.

3) Авторизуватися під обліковим записом адміністратора.

4) Змінити ціну існуючого товару через адмін-панель та зберегти.

– Очікуваний результат: На кроках 1 та 2 система блокує доступ і перенаправляє на головну сторінку/сторінку входу. На кроці 3 доступ надається. На кроці 4 змінена ціна миттєво відображається в публічному каталозі.

Метод тестування користувацького інтерфейсу (UI/UX) та продуктивності на клієнті:

Цей метод використовується для оцінки чуйності застосунку та якості кешування даних.

Сценарій 1: Адаптивність (Responsive Design).

– Дії: Змінити розмір вікна браузера до ширини мобільного екрана (375px).

– Очікуваний результат: Сітка карток товарів (Grid) перебудовується в одну колонку, головне навігаційне меню ховається за іконкою «гамбургер», розміри шрифтів адаптуються. Немає горизонтальної прокрутки.

Сценарій 2: Робота кешування (TanStack React Query).

– Дії: Завантажити каталог, перейти на сторінку товару, а потім натиснути кнопку «Назад» у браузері.

– Очікуваний результат: Сторінка каталогу відкривається миттєво, без повторного відображення індикаторів завантаження (спінерів), оскільки дані беруться з кешу, поки відбувається фонова перевірка актуальності.

Метод тестування обробки помилок та стійкості системи (Negative Testing):

Тест-кейс 6: Валідація форм клієнтською частиною (Zod).

– Опис: Перевірка захисту від введення некоректних даних.

– Кроки: У формі підбору сумісності ввести від'ємне значення (наприклад, -50 Вт). У формі оформлення замовлення залишити обов'язкові поля порожніми або ввести email без символу «@».

– Очікуваний результат: Бібліотека Zod перехоплює введення. Кнопки відправки форми блокуються. Під відповідними полями з'являються червоні повідомлення про помилку валідації. HTTP-запит на сервер не відправляється.

Тест-кейс 7: Обробка відсутності товару на складі.

– Опис: Перевірка логіки обмеження покупок.

– Кроки: Знайти товар, залишок якого на складі дорівнює 2 шт. Спробувати змінити кількість у кошику на 3 шт.

– Очікуваний результат: Система не дозволяє збільшити лічильник вище актуального залишку. Користувач отримує toast-сповіщення: «Досягнуто максимальної кількості товару на складі».

Тест-кейс 8: Стійкість до втрати зв'язку із сервером.

– Опис: Перевірка реакції застосунку на падіння REST API (Node.js).

– Кроки: Штучно зупинити серверний процес (або вимкнути інтернет). Спробувати оновити сторінку каталогу.

– Очікуваний результат: Застосунок не "падає" з фатальною помилкою (білий екран). Спрацьовує React Error Boundary (або обробник помилок React Query), який відображає користувачеві ввічливе повідомлення: «Не вдалося завантажити дані. Перевірте з'єднання або спробуйте пізніше» та пропонує кнопку для повторної спроби запиту.

ДОДАТОК Г – Керівництво користувача (оператора) вебзастосунку «V-Core»

1 Призначення програми

Експлуатаційне призначення програмного забезпечення для автоматизації обробки онлайн-замовлень та підбору відеокарт для магазину «V-Core» полягає в автоматизації та оптимізації процесів обліку товарів у каталозі, оформлення продажів, управління складськими залишками та взаємодії з клієнтами. Це забезпечує ефективне управління ресурсами магазину, знижує ймовірність помилок при підборі технічно сумісного обладнання, підвищує швидкість обробки замовлень та полегшує роботу персоналу, роблячи процес купівлі зручнішим для клієнтів.

Функціональне призначення програми: Програмне забезпечення «V-Core» (далі – вебзастосунок «V-Core») призначене для автоматизації процесів підбору та роздрібної купівлі комп'ютерних комплектуючих (відеокарт) в онлайн-режимі. Програма надає користувачам сучасний інтерактивний вебзастосунок для перегляду каталогу товарів, фільтрації за технічними характеристиками та оформлення замовлень. Ключовою особливістю є наявність модуля підбору, який автоматично перевіряє апаратну сумісність обраних графічних адаптерів із параметрами системи користувача (за рівнем енергоспоживання блоку живлення та максимальними габаритами корпусу). Крім того, програма включає захищену підсистему (панель адміністратора) для керування базою товарів та обробки замовлень.

Функціональне призначення полягає в можливості користувачів виконувати такі функції:

– реєстрація та авторизація користувачів;

– перегляд та обробка інформації про товари в каталозі (додавання, видалення, редагування технічних характеристик);

– перегляд та обробка інформації про клієнтів (додавання, видалення, редагування);

– підбір та фільтрація відеокарт за критичними параметрами (архітектура GPU, TDP, обсяг пам'яті тощо);

– створення замовлення на покупку (додавання усієї інформації щодо продажу, пошук товару та клієнта, внесення нового клієнта у разі його відсутності);

– редагування та видалення існуючого замовлення з відповідним поверненням товару на баланс складу;

– оновлення статусу замовлення (наприклад, на «виконано» або «відправлено») для подальшого відстеження;

– перегляд списку усіх товарів, доданих у конкретне замовлення клієнта;

– перегляд та видалення зареєстрованих співробітників (менеджерів);

– вихід з облікового запису.

2 Умови виконання програми

Мінімальний склад апаратних засобів:

Серверна частина (де виконується бекенд-логіка та працює СУБД):

– Процесор (vCPU): 2-ядерний, з тактовою частотою 2.0 ГГц.

– Оперативна пам’ять (RAM): 4 ГБ.

– Дисковий простір: 20 ГБ вільного місця.

Клієнтська частина (де користувач взаємодіє з інтерфейсом):

– Будь-який персональний комп’ютер, ноутбук, планшет або смартфон, здатний запустити сучасний веббраузер.

– Підключення до мережі Інтернет для доступу до сервера.

– Пристрої вводу-виводу: клавіатура, маніпулятор типу «миша» (або сенсорний екран), монітор.

Мінімальний склад програмних засобів:

Серверна частина:

– Операційна система: Linux (рекомендовано), macOS або Windows.

– Середовище виконання: Node.js версії 18 або новішої.

– Система управління базами даних (СУБД): SQLite версії 8.0 або новішої.

– Менеджер пакетів: npm (або pnpm/yarn).

Клієнтська частина:

– Сучасний веббраузер з підтримкою HTML5, CSS3 та JavaScript стандарту ES6+ (наприклад, Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge останніх версій).

Вимоги до персоналу (користувачів):

– Для клієнтів (покупців): Користувачі повинні мати базові навички роботи з персональним комп’ютером або смартфоном та веббраузером. Спеціальної технічної підготовки для роботи з інтерфейсом вебзастосунку «V-Core» не потрібно. Для ефективного використання модуля сумісності бажано знати базові характеристики власного ПК (зокрема, потужність встановленого блоку живлення).

– Для контент-менеджерів (робота в панелі адміністратора): Навички впевненого користувача ПК, загальне розуміння асортименту комп'ютерних комплектуючих та специфікацій відеокарт для коректного заповнення товарного каталогу.

– Для системних адміністраторів (розгортання та підтримка серверної частини): Навички роботи з командним рядком (терміналом), базові знання адміністрування серверних ОС (Linux), розуміння принципів роботи вебсерверів (Nginx), навички конфігурації СУБД SQLite та роботи з екосистемою Node.js.

3 Виконання програми

Завантаження та запуск програми:

Серверна частина:

– Переконайтеся, що всі необхідні програмні компоненти (середовище Node.js, СУБД SQLite) встановлені на сервері.

– Переконайтеся, що сервер бази даних SQLite запущений та налаштований (створена база даних магазину vcore_db та відповідний користувач).

– За потреби виконайте міграції та заповніть базу даних початковими тестовими даними (каталог відеокарт, тестовий адміністратор).

– Перейдіть у директорію бекенд-частини проєкту через командний рядок.

– Встановіть необхідні залежності командою npm install.

– Запустіть серверний додаток (REST API). Типова команда для локального запуску: npm run dev (або node server.js / pm2 start server.js для продакшену).

Клієнтська частина:

– Якщо вебзастосунок запускається локально для розробки, перейдіть у директорію фронтенду та виконайте команду npm run dev (інструмент Vite запустить локальний сервер розробки).

– Відкрийте сучасний веббраузер на пристрої користувача.

– В адресному рядку браузера введіть URL-адресу клієнтського застосунку. Наприклад, при локальному запуску адреса буде http://localhost:5173. Якщо вебзастосунок розгорнуто в мережі Інтернет, вкажіть його доменне ім’я (наприклад, https://vcore-shop.com).

– Після завантаження сторінки відобразиться головний інтерфейс вебзастосунку «V-Core».

Виконання програми (робота з інтерфейсом): Головний інтерфейс вебзастосунку «V-Core» надає користувачам наступні можливості:

Перегляд каталогу та використання фільтрів:

– Перейдіть до розділу «Каталог».

– На панелі зліва (або у випадаючому меню на мобільних пристроях) оберіть бажані параметри: виробник графічного чипа (NVIDIA/AMD), мінімальний/максимальний обсяг відеопам'яті, серію або ціновий діапазон.

– Список карток товарів миттєво оновиться, відображаючи лише ті моделі, які відповідають обраним критеріям.

Підбір відеокарти (перевірка сумісності):

– Перейдіть до розділу «Підбір відеокарти».

– У поле «Потужність блоку живлення» введіть значення у ватах (наприклад, 600) або скористайтеся повзунком.

– У поле «Максимальна довжина GPU» введіть доступне місце у вашому корпусі в міліметрах (наприклад, 300).

– Натисніть кнопку «Підібрати». Система автоматично відсіє моделі, які вимагають більшої потужності або не помістяться в корпус.

Оформлення замовлення:

– На картці обраного товару натисніть кнопку «Додати в кошик».

– Натисніть на іконку кошика у верхній навігаційній панелі.

– За потреби змініть кількість товарів за допомогою кнопок «-» та «+». Загальна вартість перерахується автоматично.

– Натисніть кнопку «Оформити замовлення». Заповніть форму з контактними даними (ПІБ, телефон, адреса відділення пошти) та підтвердіть покупку.

Робота в панелі адміністратора:

– Перейдіть на сторінку авторизації та введіть облікові дані адміністратора (для тестового доступу: логін – admin, пароль – admin).

– Після успішного входу в навігаційному меню з'явиться розділ «Адмін-панель».

– У вкладці «Товари» адміністратор може додавати нові відеокарти в базу даних, редагувати їх технічні характеристики, змінювати ціну та статус наявності.

– У вкладці «Коментарі» доступний інтерфейс модерації відгуків користувачів із можливістю видалення спаму.

Завершення роботи програми:

Клієнтська частина:

– Для завершення сеансу роботи в ролі авторизованого користувача або адміністратора необхідно натиснути кнопку «Вийти» (Logout) в особистому кабінеті, щоб очистити токени доступу.

– Для загального завершення роботи з вебзастосунком «V-Core» достатньо закрити вкладку або вікно веббраузера.

Серверна частина:

– Для зупинки серверного API необхідно зупинити процес Node.js у терміналі (зазвичай це робиться натисканням комбінації клавіш Ctrl+C). У випадку використання менеджера процесів (PM2) на бойовому сервері, необхідно виконати команду зупинки (наприклад, pm2 stop vcore-api).

ДОДАТОК Д – Лістинги ключових програмних модулів

Д.1 src/data/gpuData.ts

```
/**
 * Interface representing a Graphics Processing Unit (GPU) entity.
 */
export interface GPU {
  id: string; // Unique identifier for the GPU
  name: string; // Full commercial name
  brand: "NVIDIA" | "AMD"; // Manufacturer
  series: string; // Product family (e.g., RTX 40, RX 7000)
  architecture: string; // Microarchitecture (e.g., Ada Lovelace, RDNA 3)
  memoryGB: number; // VRAM capacity in Gigabytes
  memoryType: string; // VRAM technology (e.g., GDDR6X)
  coreClock: number; // Base clock speed in MHz
  boostClock: number; // Maximum boost clock speed in MHz
  tdpWatts: number; // Thermal Design Power in Watts
  lengthMM: number; // Physical length for case compatibility check
  interface: string; // Connection interface (e.g., PCIe 4.0 x16)
  outputs: string[]; // Available video outputs
  price: number; // Price in local currency
  inStock: boolean; // Availability status
  stockCount: number; // Number of units available
  imageUrl?: string; // Optional path to product image
  rating: number; // User or benchmark rating (out of 5.0)
}

/**
 * Empty catalog as placeholder. Data is now fetched from the database.
```

```
 */
export const gpuCatalog: GPU[] = [];

// Helper constants will be calculated dynamically in components or context
export const allBrands: string[] = [];
export const allArchitectures: string[] = [];
export const allMemoryTypes: string[] = [];
export const allMemorySizes: number[] = [];
```

Д.2 src/pages/SelectorPage.tsx

```
import React, { useState, useMemo, useCallback } from "react";
import { useGpus } from "@/context/GpuContext";
import GpuCard from "@/components/GpuCard";
import { Zap, Monitor, Gamepad2, Cpu, ArrowDownAZ, ArrowUpAZ } from
"lucide-react";

interface SystemConfig {
  psuWatts: number;
  caseMaxLengthMM: number;
  usage: "gaming" | "creative" | "office";
  budget: number;
  sortKey: string;
  sortOrder: "asc" | "desc";
}

const SelectorPage = () => {
  const { gpus } = useGpus();
  const [config, setConfig] = useState<SystemConfig>({
    psuWatts: 1200,
    caseMaxLengthMM: 400,
```

```
    usage: "gaming",
    budget: 80000,
    sortKey: "price",
    sortOrder: "asc",
  });

  /**
   * Memoized logic for filtering and sorting recommendations based on system configuration.
   * This is the core logic for the GPU selection algorithm.
   */
  const recommendations = useMemo(() => {
    let list = gpus.filter(gpu => {
      // PSU Compatibility: GPU TDP + overhead (approx 200W) must be within PSU capacity
      if (gpu.tdpWatts + 200 > config.psuWatts) return false;
      // Physical Fit: GPU length must not exceed the maximum allowed by the PC case
      if (gpu.lengthMM > config.caseMaxLengthMM) return false;
      // Budget Constraint: Only include cards within the specified price range
      if (gpu.price > config.budget) return false;
      // Inventory Check: Ensure the card is currently available
      if (!gpu.inStock) return false;
      return true;
    });

    /**
     * Sorting logic based on selected criteria
     */
    list.sort((a, b) => {
```

```
    let comparison = 0;
    switch (config.sortKey) {
      case "price":
        comparison = a.price - b.price;
        break;
      case "memory":
        comparison = a.memoryGB - b.memoryGB;
        break;
      case "tdp":
        comparison = a.tdpWatts - b.tdpWatts;
        break;
      case "rating":
        comparison = a.rating - b.rating;
        break;
      case "usage":
        // Context-aware recommendation sorting based on user's primary use case
        if (config.usage === "gaming") comparison = b.boostClock - a.boostClock;
        else if (config.usage === "creative") comparison = b.memoryGB -
a.memoryGB;
        else comparison = a.price - b.price;
        break;
    }
    return config.sortOrder === "asc" ? comparison : -comparison;
  });

  return list;
 }, [config, gpus]);

 /**
```

```
   * Identifies GPUs that are incompatible due to hardware constraints (PSU or Case size).
   */
  const incompatible = useMemo(() => {
    return gpus.filter(gpu => {
      if (gpu.tdpWatts + 200 > config.psuWatts) return true;
      if (gpu.lengthMM > config.caseMaxLengthMM) return true;
      return false;
    });
  }, [config.psuWatts, config.caseMaxLengthMM, gpus]);

  /**
   * Helper function to update state with partial configuration objects.
   */
  const updateConfig = useCallback((partial: Partial<SystemConfig>) => {
    setConfig(prev => ({ ...prev, ...partial }));
  }, []);

  return (
    <div className="container mx-auto px-4 py-8">
      <h1 className="mb-2 font-display text-4xl font-black uppercase tracking-tight text-foreground leading-tight">
        Підбір <span className="text-primary neon-text">відеокарт</span>
      </h1>
      <p className="mb-8 font-body text-muted-foreground">
        Вкажіть параметри вашої системи, і ми підберемо сумісні відеокарти
      </p>

      <div className="grid gap-8 lg:grid-cols-[320px_1fr]">
```

```
      <div className="space-y-6 rounded-lg border border-border bg-card p-5 neon-border h-fit">
        <h3 className="font-display text-sm font-bold uppercase tracking-wider text-primary">
          Параметри системи
        </h3>

        {/* PSU */}
        <div>
          <label className="mb-2 flex items-center gap-2 font-body text-xs font-bold uppercase tracking-widest text-muted-foreground">
            <Zap className="h-3.5 w-3.5 text-neon-orange" />
            Блок живлення: {config.psuWatts} Вт
          </label>
          <input
            type="range"
            min={400}
            max={1200}
            step={50}
            value={config.psuWatts}
            onChange={e => updateConfig({ psuWatts: Number(e.target.value) })}
            className="w-full accent-primary"
          />
          <div className="flex justify-between font-body text-[10px] text-muted-foreground">
            <span>400 Вт</span><span>1200 Вт</span>
          </div>
        </div>

        {/* Case length */}
```

```
        <div>
          <label className="mb-2 flex items-center gap-2 font-body text-xs font-bold
uppercase tracking-widest text-muted-foreground">
            <Monitor className="h-3.5 w-3.5 text-primary" />
            Макс. довжина GPU: {config.caseMaxLengthMM} мм
          </label>
          <input
            type="range"
            min={200}
            max={400}
            step={10}
            value={config.caseMaxLengthMM}
            onChange={e => setConfig(c => ({ ...c, caseMaxLengthMM:
Number(e.target.value) }))}
            className="w-full accent-primary"
          />
          <div className="flex justify-between font-body text-[10px] text-muted-
foreground">
            <span>200 мм</span><span>400 мм</span>
          </div>
        </div>

        {/* Usage */}
        <div>
          <label className="mb-2 block font-body text-xs font-bold uppercase
tracking-widest text-muted-foreground">
            Тип використання
          </label>
          <div className="grid grid-cols-3 gap-2">
            {([
```

```
            { key: "gaming", label: "Ігри", icon: Gamepad2 },
            { key: "creative", label: "Творчість", icon: Cpu },
            { key: "office", label: "Офіс", icon: Monitor },
          ] as const).map(({ key, label, icon: Icon }) => (
            <button
              key={key}
              onClick={() => setConfig(c => ({ ...c, usage: key }))}
              className={`flex flex-col items-center gap-1 rounded-lg border p-3
transition-all ${
                config.usage === key
                  ? "border-primary bg-primary text-primary-foreground neon-border"
                  : "border-border text-muted-foreground hover:bg-primary hover:text-
primary-foreground"
              }`}
            >
              <Icon className="h-5 w-5" />
              <span className="font-body text-[10px] font-semibold
uppercase">{label}</span>
            </button>
          ))}
        </div>
      </div>

      {/* Budget */}
      <div>
        <label className="mb-2 block font-body text-xs font-bold uppercase
tracking-widest text-muted-foreground">
          Бюджет: {config.budget.toLocaleString("uk-UA")} ₴
        </label>
        <input
```

```
            type="range"
            min={10000}
            max={80000}
            step={5000}
            value={config.budget}
            onChange={e => setConfig(c => ({ ...c, budget: Number(e.target.value) }))}
            className="w-full accent-primary"
          />
        </div>

        {/* Sort */}
        <div>
          <label className="mb-2 block font-body text-xs font-bold uppercase
tracking-widest text-muted-foreground">
            Сортування
          </label>
          <div className="flex gap-2">
            <select
              value={config.sortKey}
              onChange={e => setConfig(c => ({ ...c, sortKey: e.target.value }))}
              className="flex-1 rounded-md border border-border bg-secondary px-3
py-2 font-body text-sm text-foreground outline-none focus:border-primary"
            >
              <option value="usage">Рекомендовано</option>
              <option value="price">Ціна</option>
              <option value="memory">Пам'ять</option>
              <option value="tdp">TDP</option>
              <option value="rating">Рейтинг</option>
            </select>
            <button
```

```
            onClick={() => setConfig(c => ({ ...c, sortOrder: c.sortOrder === "asc" ? "desc" : "asc" }))}
            className="flex h-10 w-10 items-center justify-center rounded-md border border-border bg-secondary text-foreground hover:bg-primary hover:text-primary-foreground"
            title={config.sortOrder === "asc" ? "По зростанню" : "По спаданню"}
          >
            {config.sortOrder === "asc" ? <ArrowUpAZ size={20} /> : <ArrowDownAZ size={20} />}
          </button>
        </div>
      </div>

      {/* Compatibility warnings */}
      {incompatible.length > 0 && (
        <div className="rounded-md bg-neon-orange/10 p-3">
          <p className="font-body text-xs text-neon-orange">
            {incompatible.length} відеокарт несумісні з вашою системою (TDP або габарити)
          </p>
        </div>
      )}
    </div>

    <div>
      <p className="mb-4 font-mono text-sm font-bold text-muted-foreground uppercase tracking-widest">
        Рекомендовано: {recommendations.length}
      </p>
      {recommendations.length === 0 ? (
```

```
          <div className="flex h-64 items-center justify-center rounded-lg border
border-border bg-card">
            <p className="text-center font-body text-muted-foreground whitespace-
pre-line">
              Немає сумісних відеокарт. Спробуйте збільшити потужність БЖ або
бюджет.
            </p>
          </div>
        ) : (
          <div className="grid gap-5 sm:grid-cols-2 xl:grid-cols-3">
            {recommendations.map(gpu => (
              <GpuCard key={gpu.id} gpu={gpu} />
            ))}
          </div>
        )}
      </div>
    </div>
  </div>
 );
};

export default SelectorPage;
```

Д.3 src/pages/GpuDetailsPage.tsx

```
import { useParams, useNavigate, Link } from "react-router-dom";
import { useGpus } from "@/context/GpuContext";
import { useAuth } from "@/context/AuthContext";
import { useComments } from "@/context/CommentsContext";
import { useCart } from "@/context/CartContext";
import { useCompare } from "@/context/CompareContext";
```

```
import {
  ChevronLeft,
  ShoppingCart,
  Zap,
  MemoryStick,
  Gauge,
  ShieldCheck,
  Cpu,
  Box,
  Monitor,
  CheckCircle2,
  GitCompare,
  Trash2,
  Send,
  User,
  Star
} from "lucide-react";
import { motion } from "framer-motion";
import { toast } from "sonner";
import { Button } from "@/components/ui/button";
import { Badge } from "@/components/ui/badge";
import { Separator } from "@/components/ui/separator";
import { Textarea } from "@/components/ui/textarea";
import { useState, useEffect } from "react";
import StarRating from "@/components/StarRating";

const GpuDetailsPage = () => {
  const { id } = useParams();
  const navigate = useNavigate();
  const { getGpuById } = useGpus();
```

```
  const { user, isAdmin, isAuthenticated } = useAuth();
  const { getCommentsByGpuId, addComment, deleteComment, fetchGpuComments,
loading: commentsLoading } = useComments();
  const { addToCart } = useCart();
  const { addToCompare, removeFromCompare, isInCompare, compareList } =
useCompare();
  const [commentText, setCommentText] = useState("");
  const [rating, setRating] = useState(5);

  const gpu = getGpuById(id || "");

  useEffect(() => {
    if (id) {
      fetchGpuComments(id);
    }
  }, [id, fetchGpuComments]);

  const comments = getCommentsByGpuId(id || "");

  /**
   * Calculate average rating from user comments.
   * If no comments exist, fall back to the default product rating.
   */
  const ratedComments = comments.filter(c => c.rating !== undefined);
  const avgRating = ratedComments.length > 0
    ? ratedComments.reduce((acc, curr) => acc + (curr.rating || 0), 0) /
ratedComments.length
    : gpu?.rating || 0;

  if (!gpu) {
```

```
  return (
    <div className="container mx-auto px-4 py-20 text-center">
      <h2 className="mb-4 text-2xl font-bold">Відеокарта не знайдена</h2>
      <Button onClick={() => navigate("/catalog")}>Каталог</Button>
    </div>
  );
}

/**
 * Adds the current GPU to the shopping cart.
 */
const handleAdd = () => {
  addToCart(gpu);
  toast.success(`${gpu.name} додано до кошика`);
};

/**
 * Toggles the GPU in the comparison list.
 * Limits comparison to 4 items.
 */
const toggleCompare = () => {
  if (isInCompare(gpu.id)) {
    removeFromCompare(gpu.id);
    toast.info(`${gpu.name} Вилучено з порівняння`);
  } else {
    if (compareList.length >= 4) {
      toast.error("Можна порівняти до 4 відеокарт");
      return;
    }
    addToCompare(gpu.id);
```

```
      toast.success(`${gpu.name} Додано до порівняння`);
    }
  };

  /**
   * Configuration for technical specifications grid.
   */
  const specs = [
    { icon: Cpu, label: "Архітектура", value: gpu.architecture },
    { icon: MemoryStick, label: "Пам'ять", value: `${gpu.memoryGB} ГБ
${gpu.memoryType}` },
    { icon: Zap, label: "TDP", value: `${gpu.tdpWatts} Вт` },
    { icon: Gauge, label: "Частота", value: `${gpu.coreClock} / ${gpu.boostClock}
МГц` },
    { icon: Box, label: "Довжина", value: `${gpu.lengthMM} мм` },
    { icon: Monitor, label: "Інтерфейс", value: gpu.interface },
  ];

  return (
    <div className="container mx-auto px-4 py-8">
      <Button
        variant="ghost"
        onClick={() => navigate(-1)}
        className="mb-6 flex items-center gap-2 text-muted-foreground hover:text-
foreground"
      >
        <ChevronLeft className="h-4 w-4" /> Назад
      </Button>

      <div className="grid gap-8 lg:grid-cols-2">
```

```
{/* Image Section */}
<motion.div
 initial={{ opacity: 0, x: -20 }}
 animate={{ opacity: 1, x: 0 }}
 className="relative flex items-center justify-center rounded-xl border border-border bg-card p-12 neon-border"
>
 <Badge
  className={`absolute right-4 top-4 font-display text-xs font-bold uppercase tracking-wider ${
   gpu.brand === "NVIDIA" ? "bg-neon-green/10 text-neon-green" : "bg-neon-red/10 text-neon-red"
  }`}
 >
  {gpu.brand}
 </Badge>
 {gpu.imageUrl ? (
  <img
   src={gpu.imageUrl}
   alt={gpu.name}
   className="h-auto max-w-full object-contain"
  />
 ) : (
  <div className="flex flex-col items-center gap-4 text-muted-foreground/20">
   <Cpu className="h-32 w-32" />
   <p className="font-display font-bold">Зображення відсутнє</p>
  </div>
 )}
</motion.div>
```

```
        {/* Info Section */}
        <motion.div
          initial={{ opacity: 0, x: 20 }}
          animate={{ opacity: 1, x: 0 }}
          className="flex flex-col"
        >
          <div className="mb-6">
            <h1 className="mb-2 font-display text-4xl font-black text-foreground
md:text-6xl tracking-tighter leading-none">
              {gpu.name}
            </h1>
            <div className="flex items-center gap-4 mb-2">
              <div className="flex items-center gap-1.5 bg-secondary/80 backdrop-blur-
sm px-2.5 py-1 rounded-full border border-border">
                <StarRating rating={avgRating} readOnly size="sm" />
                <span className="font-display text-sm font-bold text-foreground">
                  {avgRating.toFixed(1)}
                </span>
                <span className="text-xs text-muted-foreground">
                  ({ratedComments.length > 0 ? ratedComments.length : "базовий"})
                </span>
              </div>
            </div>
            <div className="flex items-center gap-4">
              <p className="font-display text-4xl font-black text-primary neon-text
tracking-tighter">
                {gpu.price.toLocaleString("uk-UA")} ₴
              </p>
              <Badge variant={gpu.inStock ? "default" : "destructive"}>
```

```
            {gpu.inStock ? "В наявності" : "Немає"}
          </Badge>
        </div>
      </div>

      <p className="mb-8 font-body text-lg text-muted-foreground leading-relaxed">
        Професійне рішення на базі архітектури {gpu.architecture}, що забезпечує неймовірну продуктивність у {gpu.brand === "NVIDIA" ? "сучасних іграх з трасуванням променів" : "найвибагливіших завданнях"}. Оснащена {gpu.memoryGB} ГБ швидкої пам'яті {gpu.memoryType}.
      </p>

      <div className="mb-8 grid grid-cols-2 gap-4 sm:grid-cols-3">
        {specs.map((spec, i) => (
          <div key={i} className="rounded-lg border border-border bg-secondary/50 p-4 transition-colors hover:bg-secondary">
            <spec.icon className="mb-2 h-5 w-5 text-primary" />
            <p className="text-[10px] font-bold uppercase tracking-wider text-muted-foreground">
              {spec.label}
            </p>
            <p className="font-mono text-sm font-bold text-foreground">{spec.value}</p>
          </div>
        ))}
      </div>

      <div className="mb-8 rounded-lg border border-border bg-card p-4">
```

```
        <h4 className="mb-3 font-display text-xs font-bold uppercase tracking-wider text-primary">
          Виходи та інтерфейси
        </h4>
        <div className="flex flex-wrap gap-2">
          {gpu.outputs.map((out, i) => (
            <Badge key={i} variant="outline" className="flex items-center gap-1.5 border-primary/20 bg-primary/5">
              <CheckCircle2 className="h-3 w-3 text-neon-green" /> {out}
            </Badge>
          ))}
        </div>
      </div>

      <div className="mt-auto flex flex-col gap-4 sm:flex-row">
        <Button
          size="lg"
          onClick={handleAdd}
          disabled={!gpu.inStock}
          className="flex-1 gap-2 bg-primary font-bold uppercase tracking-wider hover:neon-box"
        >
          <ShoppingCart className="h-5 w-5" /> Додати у кошик
        </Button>
        <Button
          variant="outline"
          size="lg"
          onClick={toggleCompare}
          className={`flex-1 gap-2 border-primary transition-colors ${
            isInCompare(gpu.id)
```

```
              ? "bg-primary text-primary-foreground neon-box"
              : "text-primary hover:bg-primary hover:text-primary-foreground"
            }`}
          >
            <GitCompare className="h-5 w-5" /> {isInCompare(gpu.id) ? "Вже у порівнянні" : "Додати до порівняння"}
          </Button>
          <Link to="/selector" className="flex-1">
            <Button
              variant="outline"
              size="lg"
              className="w-full gap-2 border-primary text-primary transition-colors hover:bg-primary hover:text-primary-foreground"
            >
              <ShieldCheck className="h-5 w-5" /> Перевірити сумісность
            </Button>
          </Link>
        </div>
      </motion.div>
    </div>

    {/* Comments Section */}
    <div className="mt-16 max-w-3xl">
      <h3 className="mb-6 font-display text-2xl font-bold">Коментарі та відгуки</h3>

      {isAuthenticated ? (
        <div className="mb-8 rounded-xl border border-border bg-card p-6">
          <h4 className="mb-4 text-sm font-bold uppercase tracking-wider text-primary">Залишити відгук</h4>
```

```
          <div className="mb-4 flex items-center gap-3">
            <span className="text-sm text-muted-foreground">Ваша оцінка:</span>
            <StarRating rating={rating} onRatingChange={setRating} size="lg" />
          </div>
          <Textarea
            placeholder="Ваш відгук про цю відеокарту..."
            className="mb-4 min-h-[100px] border-primary/20 bg-background/50
focus:border-primary"
            value={commentText}
            onChange={(e) => setCommentText(e.target.value)}
          />
          <Button
            onClick={() => {
              if (commentText.trim()) {
                addComment(gpu.id, user!.id, user!.username, commentText, rating);
                setCommentText("");
                setRating(5);
                toast.success("Відгук додано");
              }
            }}
            className="gap-2"
          >
            <Send className="h-4 w-4" /> Надіслати
          </Button>
        </div>
      ) : (
        <div className="mb-8 rounded-xl border border-dashed border-border p-6
text-center">
          <p className="mb-4 text-muted-foreground">Увійдіть в аккаунт, щоб
залишити відгук</p>
```

```
          <Button variant="outline" onClick={() =>
navigate("/login")}>Увійти</Button>
        </div>
      )}

      <div className="space-y-6">
        {comments.length === 0 ? (
          <p className="text-muted-foreground italic">Поки що немає відгуків.
Будьте першим!</p>
        ) : (
          comments.map((comment) => (
            <div key={comment.id} className="group relative rounded-xl border
border-border bg-card/50 p-6 transition-colors hover:bg-card">
              <div className="mb-2 flex items-center justify-between">
                <div className="flex items-center gap-2">
                  <div className="flex h-8 w-8 items-center justify-center rounded-full
bg-primary/10 text-primary">
                    <User className="h-4 w-4" />
                  </div>
                  <div>
                    <div className="flex items-center gap-2">
                      <span className="font-bold text-
foreground">{comment.username}</span>
                      {comment.rating && (
                        <StarRating rating={comment.rating} readOnly size="sm" />
                      )}
                    </div>
                    <span className="text-[10px] text-muted-
foreground">{comment.date}</span>
                  </div>
```

```
                </div>
                {isAdmin && (
                  <Button
                    variant="ghost"
                    size="icon"
                    className="text-muted-foreground hover:text-red-500"
                    onClick={() => {
                      if (window.confirm("Видалити цей коментар?")) {
                        deleteComment(comment.id);
                        toast.success("Коментар видалено");
                      }
                    }}
                  >
                    <Trash2 className="h-4 w-4" />
                  </Button>
                )}
              </div>
              <p className="text-muted-foreground leading-
relaxed">{comment.text}</p>
            </div>
          ))
        )}
      </div>
    </div>
  </div>
 );
};

export default GpuDetailsPage;
```

ДОДАТОК Е – Діаграма класів вебзастосунку «V-Core»

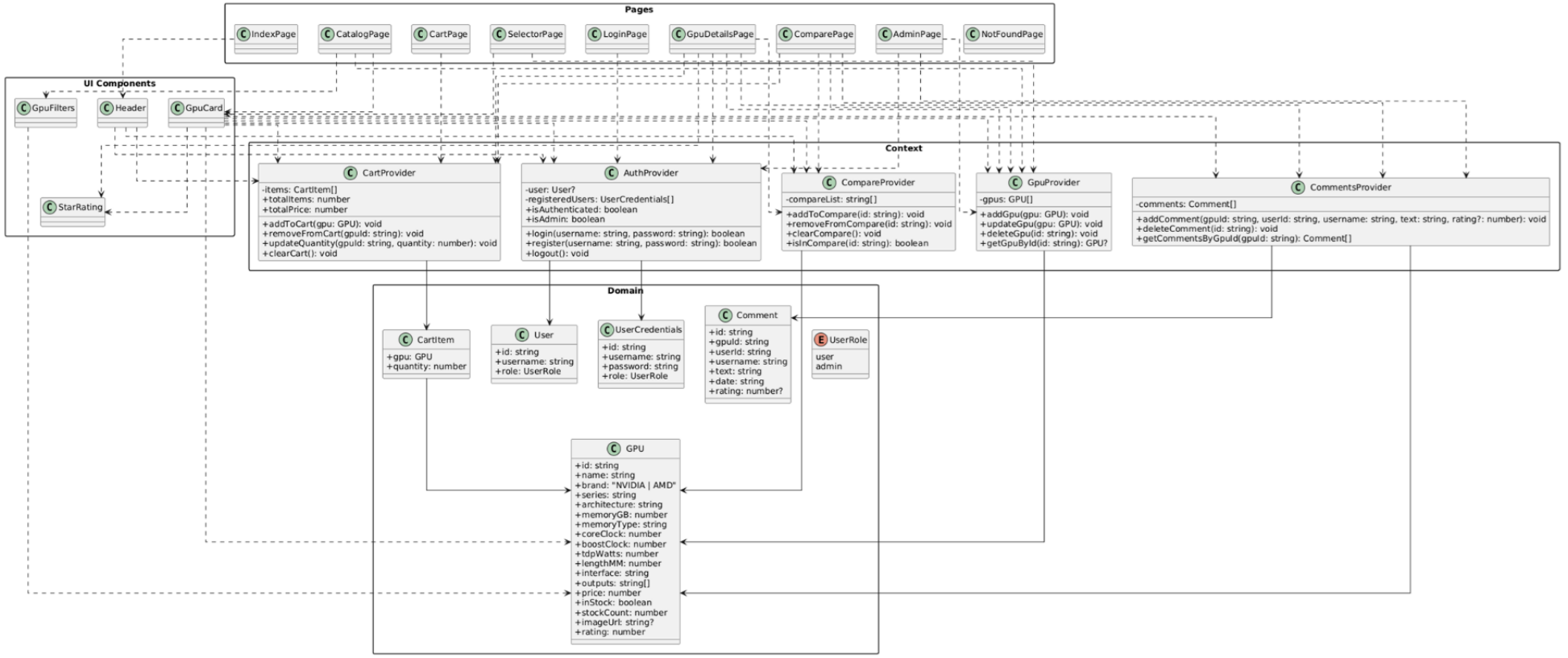


Рисунок Е.1 – Діаграма класів вебзастосунку магазину «V-Core»