Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Створення веб-сайту інтернет магазину моніторів «FrameRate»

Виконав: студент IV курсу, групи СН-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Костич А.
(підпис) (прізвище та ініціали)

Керівник _____ Небесний Р.М
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Липак Г.І.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«__» _червня_ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)
Студенту Костич Арсен
(прізвище, ім'я, по батькові)
1. Тема роботи Створення веб-сайту інтернет магазину моніторів «FrameRate»

Керівник роботи Небесний Руслан Михайлович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «*14*» *травня* 2026 року № *4/9-239*
2. Термін подання студентом завершеної роботи 22 червня 2026 р.
3. Вихідні дані до роботи Джерела з літератури та інтернету про створення клієнтської та серверної частин вебсайту.
4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1 Аналіз предметної області та постановка завдання. 1.1 Огляд існуючих рішень на ринку електронної комерції моніторів та їх аналіз 1.2 Актуальність теми та детальний Опис предметної області. 1.3 Постановка завдання розробки програмної системи «FrameRate». 1.4 Висновки до першого розділу. 2 Теоретичні основи та вимого до розробки програмної системи. 2.1 Загальні поняття та засади розробки програмного забезпечення. 2.2 Визначення підходів до розробки та методів дослідження для реалізації проєкту. 2.3 Розробка та визначення вимог до веб-платформи онлайн-магазину моніторів «FrameRate» 2.4 Обгрунтування вибору технологічного стеку та інструментальних засобів розробки вебплатформ. 2.5 Висновок до другого розділу. 3 Проєктування та реалізація інтернет магазину «FrameRate». 3.1 Архітектура та реалізація програмної системи вебсайту. 3.2 Реалізація клієнтської частини веб-додатку. 3.3 Демонстрація роботи інтерфейсу та екпериментальна верифікація результатів розробки. 3.4 Висновок до третього розділу 4 Безпека життєдіяльності, основи охорони праці. 4.1 Надання домедичної допомоги 4.2 Організація інструктажів та основні вимоги техніки безпеки під час експлуатації 4.3 Висновок до четвертого розділу. Висновки. Перелік джерел. Додаток А. Додаток Б.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Огляд існуючих рішень. 2. Актуальність обраної теми. 3. Мета, об'єкт та предмет дослідження. 4. Завдання на розробку вебсайту інтернет-магазину. 5. Теоретичні основи та вимоги до системи. 6. Діаграма варіантів використання. 7. Архітектура програмної системи. 8. Користувацький інтерфейс. 9. Корзина та процес оформлення замовлення. 10. Приклад роботи модуля інтерактивного пошуку. 11. Висновок

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання 26 січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	26.01.2026	
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	27.01.2026-16.02.2026	
3.	Виконання дослідження щодо архітектурних підходів, методів та технологій розробки сучасних систем електронної комерції	17.02.2026-10.05.2026	
4.	Розроблення веб-платформи спеціалізованого інтернет-магазину моніторів «FrameRate» на основі стеку MERN.		
5.	Оформлення розділу «Аналіз предметної області та постановка завдання»	11.05.2026-17.05.2026	
6.	Оформлення розділу «Теоретичні основи та вимоги до розробки програмного забезпечення»	18.05.2026-24.05.2026	
7.	Оформлення розділу «Проєктування та реалізація інтернет магазину «FrameRate»»	25.05.2026-31.05.2026	
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»	01.06.2026-08.06.2026	
9.	Виконання завдання до підрозділу «Основи охорони праці»	01.06.2026-08.06.2026	
10.	Оформлення кваліфікаційної роботи	09.06.2026-11.06.2026	
11.	Нормоконтроль	19.06.2026	
12.	Перевірка на плагіат	19.06.2026	
13.	Попередній захист кваліфікаційної роботи	18.06.2026	
14.	Захист кваліфікаційної роботи	23.06.2026	

Студент _____ Костич А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Небесний Р.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення веб-сайту інтернет магазину моніторів «FrameRate» // Кваліфікаційна робота освітнього ступеня «Бакалавр» // Костич Арсен // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-41 // Тернопіль, 2026 // С. 63 , рис. – 10 , табл. – 2 , кресл. – 11 , додат. – 2 , бібліогр. – 40 .

Ключові слова: електронна комерція, інтернет-магазин, стек mern, react.js, node.js, mongodb atlas, фільтрація товарів, живий пошук, single page application.

Кваліфікаційна робота присвячена дослідженню архітектурних підходів та розробці сучасних інформаційних систем електронної комерції у сегменті комп'ютерної периферії. В першому розділі кваліфікаційної роботи описано загальний аналіз предметної області роздрібної онлайн-торгівлі засобами відображення інформації, проведено детальний порівняльний огляд існуючих ринкових рішень (Rozetka, Comfy, Telemart) та сформульовано технічну постановку завдання.

В другому розділі кваліфікаційної роботи визначено повний спектр функціональних та нефункциональних вимог до платформи, побудовано UML-діаграму варіантів використання для диференціації прав доступу акторів, а також інженерно обґрунтовано вибору одномовної екосистеми на базі JavaScript-стеку MERN (MongoDB, Express.js, React.js, Node.js).

В третьому розділі кваліфікаційної роботи описано децентралізовану архітектуру програмного комплексу, структуру каталогів фронтенду та бекенду, реалізацію інтерфейсу користувача та адміністратора як Single Page Application (SPA), математичну логіку клієнтських алгоритмів живого пошуку й багаторівневої фільтрації даних, а також наведено результати експериментальної верифікації швидкодії системи.

В четвертому розділі розглянуто інженерно-технічні вимоги безпеки життєдіяльності та охорони праці розробників комп'ютерних систем.

Об’єкт дослідження: Процес веброзробки повнофункціональних систем електронної комерції (e-commerce).

Предмет дослідження: Методи, архітектурні підходи та технології створення адаптивного інтернет-магазину моніторів з використанням JavaScript-стеку.

ANNOTATION

Creation of a website for the online store of monitors «FrameRate» // Qualification work of the educational level «Bachelor» // Kostych Arsen // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-41 // Ternopil, 2026 // P. 63 , fig. – 10 , tabl. – 2 , chair. – 11 , annexes. – 2 , references – 40.

Keywords: e-commerce, online store, mern stack, react.js, node.js, mongodb atlas, product filtering, live search, single page application.

The qualification work is dedicated to the research of architectural approaches and the development of modern electronic commerce information systems in the computer peripherals segment. The first section of the qualification paper considered the general analysis of the subject area of online monitor retail, provided a detailed comparative review of existing market solutions (Rozetka, Comfy, Telemart), and formulated the technical problem statement.

In the second section of the qualification work, a full range of functional and non-functional requirements for the platform was defined, a UML Use Case diagram was constructed to differentiate actor access rights, and the choice of a single-language ecosystem based on the JavaScript MERN stack (MongoDB, Express.js, React.js, Node.js) was engineeringly justified.

The third section of the qualification work describes the decoupled architecture of the software complex, the directory structure of the frontend and backend, the implementation of the user and administrator interface as a Single Page Application (SPA), the mathematical logic of client-side live search and multi-level data filtering algorithms, and presents the results of experimental verification of system performance.

The fourth section addresses the engineering and technical requirements of occupational health and life safety of computer systems developers.

Object of research: The process of web development of full-featured electronic commerce (e-commerce) systems.

Subject of research: Methods, architectural approaches, and technologies for creating an adaptive online monitor store using the JavaScript stack

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

MIT (англ. Massachusetts Institute of Technology) – Массачусетський технологічний інститут.

ШІ – Штучний інтелект.

API (англ. Application Programming Interface) – програмний інтерфейс додатка, що забезпечує взаємодію між сервером та клієнтом.

CSS (англ. Cascading Style Sheets) – каскадні таблиці стилів для візуального оформлення веб-сторінок.

HTML (англ. HyperText Markup Language) – стандартизована мова розмітки гіпертексту для створення структури веб-документів.

JSON (англ. JavaScript Object Notation) – текстовий формат обміну даними, що використовується у веб-розробці.

ЗМІСТ

ВСТУП

Актуальність теми. Сучасний ринок комп'ютерної периферії та моніторів стрімко розвивається у зв'язку з масовим переходом на віддалену роботу, розвитком кіберспорту та підвищенням вимог до якості відображення графічного контенту. Специфіка купівлі монітора полягає в необхідності детального аналізу багатьох технічних параметрів: типу матриці (IPS, VA, OLED), частоти оновлення (60Hz, 144Hz, 240Hz, 360Hz), роздільної здатності (Full HD, 2K, 4K) та часу відгуку.

Мета і задачі дослідження. Більшість універсальних маркетплейсів мають обмежені можливості фільтрації, що ускладнює вибір для професійних дизайнерів, програмістів та геймерів. Створення спеціалізованого інтернет-магазину «FrameRate» із сучасним UX/UI, інтерактивними індикаторами характеристик та оптимізованою системою фільтрації є високоактуальним завданням для індустрії електронної комерції.

– Об'єкт дослідження: Процес веброзробки повнофункціональних систем електронної комерції (e-commerce).

– Предмет дослідження: Методи, архітектурні підходи та технології створення адаптивного інтернет-магазину моніторів з використанням JavaScript-стеку.

– Мета проєкту: Розробка швидкої, безпечної та адаптивної веб-платформи «FrameRate» для вибору та купівлі моніторів, яка мінімізує час користувача на пошук ідеальної моделі завдяки кастомним фільтрам.

Практичне значення одержаних результатів. Розроблений вебсайт «FrameRate» забезпечує зручний доступ до асортименту моніторів, полегшуючи процес вибору та купівлі завдяки інтуїтивно зрозумілому інтерфейсу, системі фільтрації товарів, кошику та зручному оформленню замовлень. Платформа дозволяє користувачам швидко знаходити потрібні моделі пристроїв, отримувати детальну інформацію про їхні технічні характеристики і

здійснювати покупки з мінімальними зусиллями. Отримані результати можуть бути використані для створення інших спеціалізованих інтернет-магазинів комп'ютерної техніки та периферії, а також у навчальних програмах із веброзробки та дизайну користувацького досвіду.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Огляд існуючих рішень на ринку електронної комерції моніторів та їх аналіз

Сучасний вітчизняний ринок електронної комерції у сегменті комп'ютерної периферії характеризується високою конкуренцією та представлений як великими універсальними маркетплейсами, так і спеціалізованими роздрібними мережами. Серед найбільш популярних онлайн-платформ в Україні, які займаються продажем засобів відображення інформації, можна виділити лідера масового маркетплейсу Rozetka [1], омніканальну мережу побутової техніки Comfy [2], а також профільний комп'ютерний центр Telemart [3]. Кожен із цих вебресурсів базується на власних архітектурних рішеннях, має унікальну структуру інтерфейсу та специфічні підходи до каталогізації товарних позицій. Проте різне ринкове позиціонування цих компаній створює суттєві відмінності у функціоналі пошуку та підбору складних технічних пристроїв, якими є сучасні монітори.

Найбільшим гравцем на ринку є онлайн-маркетплейс Rozetka, головними перевагами якого є величезна клієнтська база та звичний для мільйонів користувачів інтерфейс. Оскільки ця платформа є мультибрендовою та багатопрофільною, її система фільтрації товарів для категорії моніторів є занадто уніфікованою і позбавленою глибокої технічної деталізації. «Генералізований характер масштабних маркетплейсів, таких як Rozetka, обмежує можливість впровадження вузькоспеціалізованих пошукових фільтрів, змушуючи покупця самостійно шукати специфічні характеристики матриць у текстових описах». Крім того, залучення сторонніх продавців на платформу часто призводить до дублювання однакових моделей моніторів у каталозі з

різними назвами та фрагментарними технічними даними, що заплутує клієнта під час порівняння.

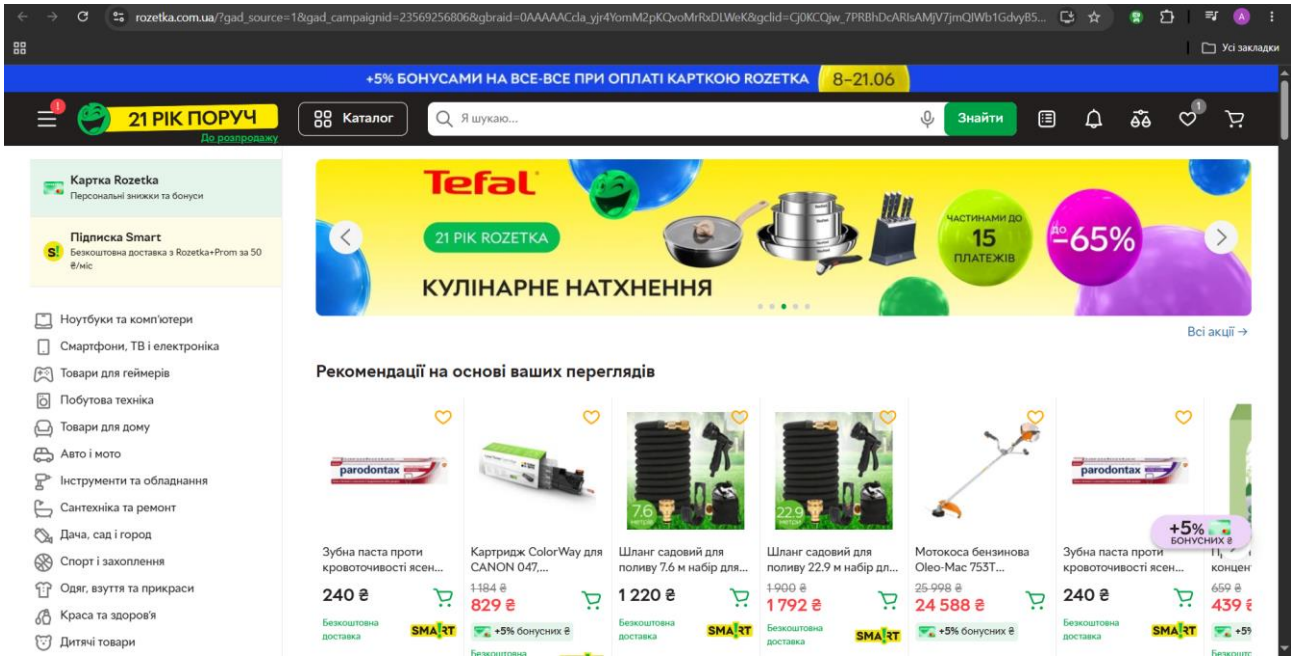

Рисунок 1.1 - Головна сторінка Rozetka

Іншим вагомим представником електронної торгівлі є мережа магазинів Comfy, яка активно розвиває свій онлайн-каталог. Вебсайт компанії має сучасний, візуально привабливий дизайн із акцентом на маркетингові пропозиції, розраховані на масового покупця. Однак специфіка цього ресурсу полягає в орієнтації на споживчий, переважно бюджетний або середньоціновий сегмент техніки, через що асортимент високотехнологічних моніторів є обмеженим. Технічні параметри дисплеїв на сайті Comfy часто скорочені до базових (діагональ, бренд, ціна), а критично важливі для професіоналів та геймерів показники, такі як колірне охоплення (sRGB, DCI-P3), точний час відгуку (GtG) чи підтримка технологій адаптивної синхронізації (G-Sync, FreeSync), взагалі відсутні в системі фільтрів.

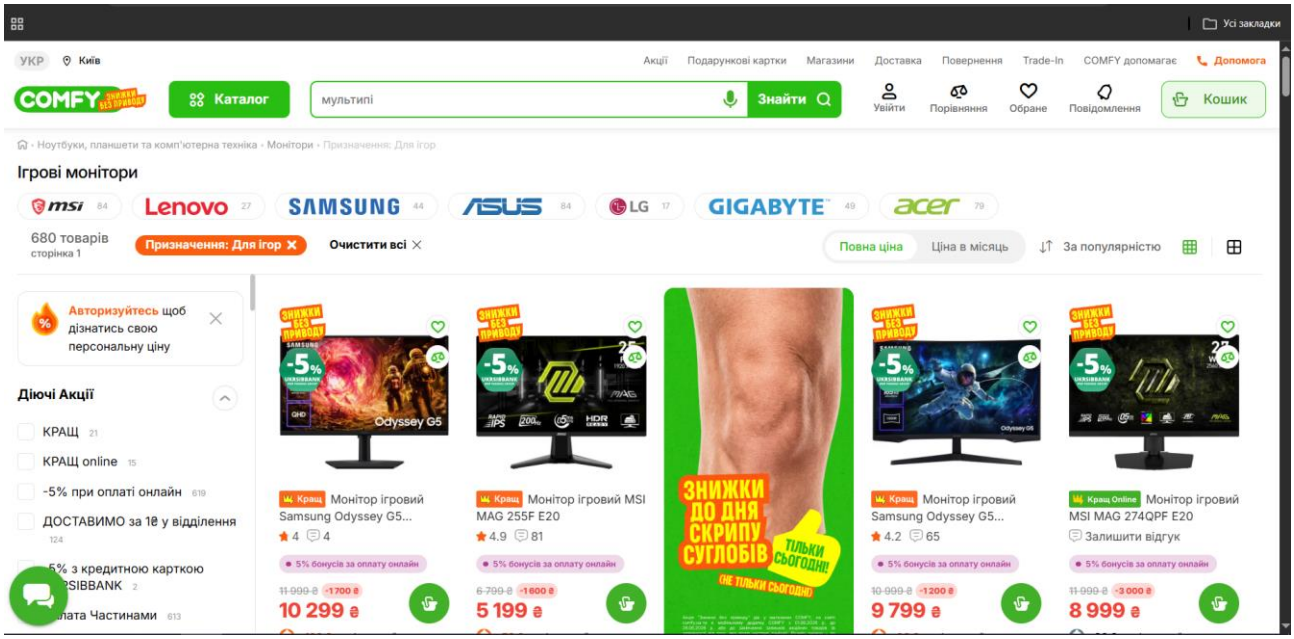

Рисунок 1.2 - Головна сторінка Comfy

На відміну від вищезгаданих ритейлерів, спеціалізований інтернет-магазин Telemart орієнтований на технічно підковану аудиторію — геймерів, кіберспортсменів та ПК-ентузіастів. Вебсайт платформи пропонує дійсно глибоку систему фільтрації моніторів за специфічними геймерськими та професійними параметрами, включаючи частоту оновлення екрана (від 144 Hz до 360 Hz) та типи відеовходів. Проте такий підхід породив іншу проблему: інтерфейс каталогу Telemart є критично перевантаженим складною термінологією, важкими для сприйняття таблицями та щільними інформаційними блоками, що відлякує та дезорієнтує пересічних або недосвідчених користувачів. Також динамічне оновлення сторінки при виборі багатьох фільтрів одночасно демонструє помітні затримки через важкі синхронні запити до бази даних.

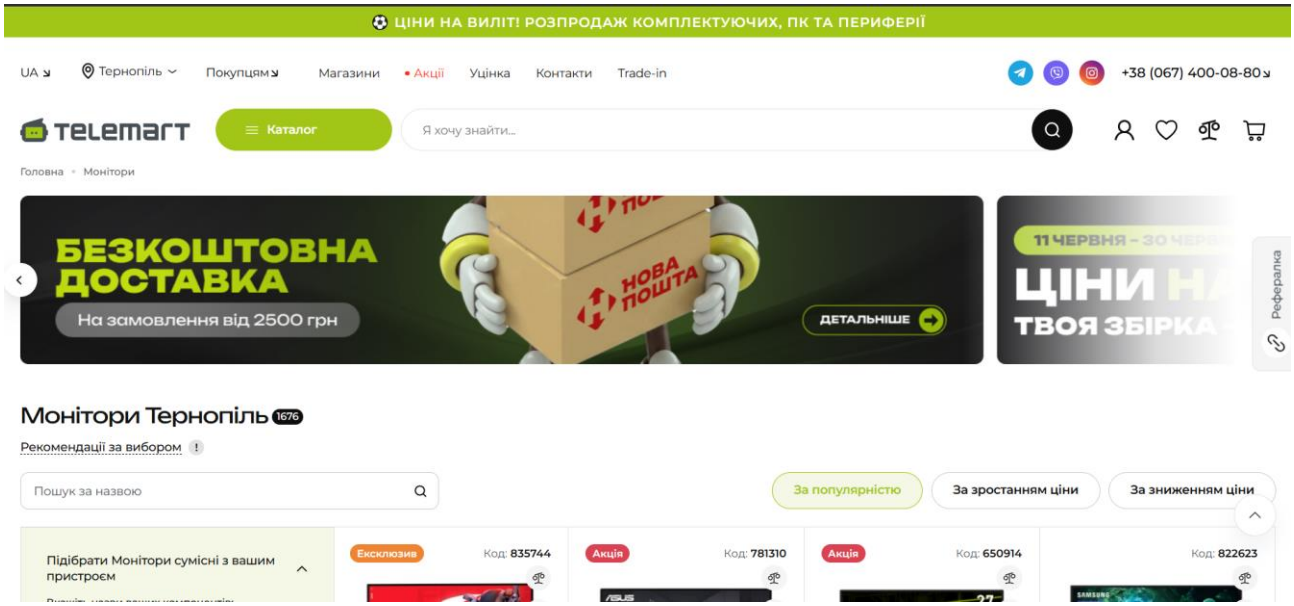

Рисунок 1.3 - Головна сторінка Telemart

Проведений детальний аналіз існуючих рішень дозволив виявити низку суттєвих недоліків: великі платформи (Rozetka, Comfy) страждають від браку технічної глибини та точних фільтрів, тоді як профільні сайти (Telemart) є занадто складними для масового споживача і мають перевантажений інтерфейс. Для усунення цих мінусів розробляється спеціалізований інтернет-магазин моніторів "FrameRate", який поєднає в собі найкращі практики конкурентів. Створюваний вебресурс запропонує користувачам збалансований UX/UI-інтерфейс із інтуїтивним поділом фільтрів за категоріями використання (для геймінгу, офісу чи дизайну) та водночас збереже максимальну точність технічних специфікацій. Завдяки використанню сучасного стеку технологій MERN, «FrameRate» забезпечить миттєву динамічну фільтрацію без перезавантаження сторінок, гарантуючи високу швидкість роботи та комфорт для будь-якого клієнта.

1.2 Актуальність теми та детальний опис предметної області

Стрімкий розвиток цифрових технологій, масовий перехід компаній на віддалену або гібридну форму роботи, а також глобальне зростання індустрії

кіберспорту та створення мультимедійного контенту зумовили значне підвищення вимог користувачів до засобів відображення інформації. Сучасний монітор перестав бути просто другорядним периферійним пристроєм для виведення текстових даних; він перетворився на складний багатофункціональний інструмент, від інженерних характеристик якого безпосередньо залежить продуктивність праці розробника, точність роботи дизайнера, результативність геймера та загальний стан здоров'я очей людини. З огляду на це, попит на якісні комп'ютерні дисплеї щороку демонструє стабільне зростання, що робить розробку та впровадження спеціалізованих вебплатформ для їхньої дистрибуції високоактуальним завданням для сучасного e-commerce сектору України [4].

Предметна область проєкту охоплює широке коло специфічних технічних параметрів, які безпосередньо визначають сценарії експлуатації моніторів та формують структуру бази даних майбутнього сайту. Ключовими характеристиками дисплеїв є тип матриці (IPS, VA, OLED), частота оновлення екрана (від стандартних 60Hz для офісу до преміальних 240-360 Hz для кіберспорту), роздільна здатність FullHD, 2K, 4K, 5K), час відгуку пікселя в мілісекундах, а також типи інтерфейсів підключення (HDMI, DisplayPort, Type-C із підтримкою Power Delivery) [5]. «Процес вибору монітора є складним багатофакторним завданням, оскільки різні категорії користувачів висувають взаємовиключні вимоги до обладнання: графічним дизайнерам необхідне максимальне колірне охоплення sRGB чи DCI-P3, тоді як геймерам — мінімальний input lag та підтримка технологій адаптивної синхронізації екрана» [6].

У зв'язку з такою глибокою диференціацією характеристик, детальний опис предметної області інтернет-магазину "FrameRate" вимагає чіткої сегментації товарного асортименту за цільовим призначенням. Традиційні маркетплейси часто згруповують техніку виключно за брендами або ціною, що змушує покупця самостійно вивчати складні інженерні специфікації. Спеціалізована вебплатформа має оперувати концепцією готових рішень, де

кожна модель монітора чітко класифікується за категоріями використання: для професійної роботи з кольором, динамічного геймінгу чи повсякденних офісних завдань. Це вимагає створення гнучкої моделі даних NoSQL на серверній частині, здатної ефективно обробляти специфічні властивості товарів без втрати швидкодії системи [7].

Таким чином, актуальність створення інтернет-магазину «FrameRate» полягає в об'єктивній потребі ринку в нішевому вебресурсі, який трансформує складні апаратні характеристики сучасних дисплеїв у зрозумілий, інтуїтивний та структурований інтерфейс користувача. Побудова детальної інформаційної моделі предметної області у поєднанні з інтерактивними інструментами порівняння[14] та кастомної фільтрації дозволить не лише оптимізувати процес підбору монітора для клієнта, а й автоматизувати внутрішні процеси обліку, контент-менеджменту та обробки замовлень на стороні адміністратора розроблюваної системи.

1.3 Постановка завдання розробки програмної системи «FrameRate»

На основі проведеного аналізу предметної області, дослідження ринкових аналогів та виявлених недоліків існуючих рішень, головною метою цієї кваліфікаційної роботи є проєктування, програмна реалізація та тестування архітектури спеціалізованого інтернет-магазину моніторів «FrameRate". Створювана система має забезпечити оптимальний баланс між глибокою технічною деталізацією характеристик дисплеїв (тип матриці, частота оновлення, час відгуку) та інтуїтивно зрозумілим, неперевантаженим інтерфейсом користувача. Для досягнення високої швидкодії, миттєвої динамічної фільтрації та масштабованості системи як технологічний стек обрано MERN (MongoDB, Express.js, React.js, Node.js).

Для реалізації поставленої мети в межах розробки програмної системи необхідно вирішити такі ключові завдання:

– Проєктування гнучкої нереляційної структури бази даних NoSQL для ефективного збереження розширених технічних специфікацій моніторів.

– Розробка серверної частини (Backend) для обробки бізнес-логіки, безпечної автентифікації користувачів та взаємодії з базою даних через REST API.

– Створення адаптивного та інтерактивного клієнтського інтерфейсу (Frontend) із реалізацією компонентів кошика, інтерактивних фільтрів та персонального кабінету.

– Інтеграція захищених механізмів авторизації на основі сесійних токенів (JWT) для розмежування прав доступу звичайних покупців та адміністраторів сайту.

Для чіткого визначення архітектурних меж майбутнього інтернет-магазину та логіки взаємодії його модулів розроблено специфікацію функціональних завдань, а також вхідних і вихідних інформаційних потоків. Деталізований опис структури програмної системи "FrameRate" наведено в таблиці 1.1.

Таблиця 1.1 — Специфікація модулів та інформаційних потоків системи "FrameRate"

Модуль системи	**Функціонал модуля**	**Вхідні дані**	**Вихідні дані**
Автентифікація	Реєстрація користувачів, вхід та захист прав доступу.	Логін, пароль, Email, ПІБ.	Акаунт у БД, токен *JWT*.
Каталог та фільтр	Пошук, сортування та кастомний підбір моніторів.	Параметри матриці (*IPS/OLED*), частоти (*Hz*).	Відфільтрований список товарів на екрані.
Кошик та замовлення	Накопичення обраних моделей та оформлення покупки.	*ID* товару, адреса доставки, телефон.	Документ замовлення в БД, оновлення складу.
Адмін-панель	Управління контентом сайту та статусами замовлень.	Характеристики моніторів, нові статуси.	Оновлена вітрина магазину та дані в *MongoDB*.

Таким чином, сформована постановка завдання чітко окреслює функціональні вимоги та логічну структуру розроблюваного вебресурсу. Реалізація зазначених у таблиці 1.1 модулів дозволить створити цілісну, стійку до навантажень систему електронної комерції. Наступним етапом проєктування

є теоретичне обґрунтування вибору архітектурних підходів та детальний аналіз вимог до платформи "FrameRate".

1.4 Висновки до першого розділу

У першому розділі кваліфікаційної роботи було проведено детальний аналіз предметної області, досліджено наявні ринкові аналоги та сформульовано технічну постановку завдання на розробку спеціалізованого інтернет-магазину моніторів "FrameRate". За результатами аналітичних досліджень можна зробити такі висновки:

Визначено високу актуальність теми дослідження, яка зумовлена стрімким розвитком комп'ютерної техніки, масовим переходом користувачів на віддалену роботу та підвищенням вимог до засобів відображення інформації. Обґрунтовано доцільність створення нішевого вебресурсу у сфері електронної комерції для оптимізації процесу підбору моніторів.

Проведено порівняльний аналіз великих українських онлайн-платформ, таких як маркетплейс Rozetka, торговельна мережа Comfy та профільний комп'ютерний центр Telemart. Виявлено їхні основні недоліки: універсальні сайти страждають від браку технічної глибини фільтрів, тоді як вузькоспеціалізовані ресурси мають перевантажений і складний для пересічного покупця інтерфейс.

Сформовано детальний опис предметної області інтернет-магазину. Виділено ключові апаратні характеристики сучасних дисплеїв (типи матриць IPS, VA, OLED, частота оновлення в Hz, роздільна здатність, час відгуку пікселя), які стали основою для подальшого проєктування структури бази даних NoSQL та побудови системи кастомної фільтрації за цільовим призначенням[15] (геймінг, дизайн, офіс).

Сформульовано чітку постановку завдання на розробку інформаційної системи "FrameRate" на основі сучасного асинхронного JavaScript-стеку MERN

(MongoDB, Express.js, React.js, Node.js). Визначено межі проєкту, виділено основні функціональні модулі платформи (автентифікація, каталог, кошик, адмін-панель) та розроблено специфікацію вхідних і вихідних інформаційних потоків.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ОСНОВИ ТА ВИМОГИ ДО РОЗРОБКИ ПРОГРАМНОЇ СИСТЕМИ

2.1 Загальні поняття та засади розробки програмного забезпечення

Проєктування та інженерна реалізація сучасних систем електронної комерції (e-commerce) являє собою комплексний та багаторівневий процес, який вимагає суворого дотримання методологічних засад програмної інженерії, використання сучасних архітектурних рішень та слідування усталеним стандартам написання програмного коду. Створення спеціалізованої вебплатформи, що оперує великими масивами динамічних та сильно диференційованих даних (таких як детальні технічні специфікації моніторів, облікові записи користувачів, кошики покупців та логістичні статуси замовлень), базується на фундаментальній концепції життєвого циклу програмного забезпечення (SDLC — Software Development Life Cycle)[8]. На кожному з етапів цього циклу — від первинного системного аналізу вимог і побудови інформаційних моделей до безпосереднього написання коду, його рефакторингу, профілювання швидкодії та фінального розгортання на хмарних серверах — критично важливо мінімізувати рівень зачеплення (coupling) між окремими системними модулями та забезпечити максимальний рівень зв'язності (cohesion) всередині них.

Основними засадами, на яких будується програмна архітектура інтернет-магазину "FrameRate", є парадигми компонентно-орієнтованого веброзроблення, модульності та чіткого розподілу відповідальності (Separation of Concerns). У контексті створення сучасних вебдодатків розподіл відповідальності реалізується через тришарову архітектурну схему, де рівень представлення даних (Frontend) повністю ізольований від рівня бізнес-логіки (Backend), а останній, у свою чергу, взаємодіє з рівнем збереження даних (Database) через абстраговані інтерфейси або об'єктно-документні мапери (ODM). Такий підхід гарантує, що зміни у візуальному оформленні картки

монітора або структурі фільтрів на клієнтській частині ніяк не порушать стабільність алгоритмів обробки замовлень на сервері.

Під час безпосереднього написання програмного коду як клієнтської, так і серверної частин системи "FrameRate", ключове значення має неухильне дотримання базових інженерних принципів DRY (Don't Repeat Yourself — не повторюйся) та KISS (Keep It Simple, Stupid — роби простіше) [9]. Принцип DRY вимагає, щоб кожен елемент знань або логіки додатка мав єдине, чітке та авторитетне представлення в системі. У межах розробки нашого інтернет-магазину це досягається шляхом створення універсальних, перевикористовуваних UI-компонентів на фронтенді (наприклад, єдиний компонент картки монітора, який динамічно приймає різні пропси для відображення на головній сторінці, у каталозі чи в списку порівняння) та винесення спільних сервісних функцій валідації чи логування на бекенді в окремі проміжні програмні шари (middleware). Це не лише суттєво скорочує загальний обсяг кодової бази додатка, а й значно спрощує подальший процес усунення дефектів і модернізації системи.

У свою чергу, принцип KISS орієнтує розробника на уникнення надлишкової інженерної складності (overengineering). Замість проектування заплутаних ієрархій успадкування чи створення заважких універсальних функцій, система будується на основі простих, декларативних та легко читабельних конструкцій. Це підвищує швидкість інтерпретації коду вебсервером, оптимізує споживання оперативної пам'яті комп'ютера користувача під час рендерингу важких сторінок з десятками моделей моніторів і забезпечує легку адаптацію коду іншими програмістами у разі масштабування бізнесу та розширення асортименту товарів платформи.

2.2 Визначення підходів до розробки та методів дослідження для реалізації проєкту

Успішна реалізація повнофункціонального інтернет-магазину моніторів "FrameRate" потребує вибору ефективної методології управління процесами проектування та кодингу. Традиційна каскадна модель розробки (Waterfall), що передбачає сувору послідовність етапів без можливості повернення назад, була визнана неефективною для цього проєкту. Специфіка індустрії електронної комерції та веброзробки характеризується високою динамічністю: у процесі створення інтерфейсу системи фільтрації складних апаратних характеристик моніторів (таких як технології адаптивної синхронізації, показники колірного охоплення чи параметри підсвічування) часто виникає необхідність оперативного перегляду UX/UI-рішень на основі первинного фідбеку користувачів. З огляду на це, для реалізації проєкту було обрано гнучку методологію розробки Agile та її ітеративно-інкрементальний фреймворк Scrum [10].

Організація розробки системи в межах Scrum-підходу базується на розподілі всього обсягу запланованих робіт на фіксовані за часом цикли — спринти (sprints), тривалість яких у цьому проєкті становить два тижні. Перед початком кожного спринту на основі загального беклогу продукту (Product Backlog) формується детальний план завдань, що транслюються у форматі користувацьких історій (User Stories) [11], наприклад: «Як покупець, я хочу мати можливість відфільтрувати монітори за частотою оновлення від 144 Hz, щоб швидко знайти ігрові моделі». Наприкінці кожного ітераційного циклу створюється повністю працездатний інкремент програмного продукту[12], який проходить внутрішнє тестування та демонструється замовнику. «Впровадження гнучких ітеративних методологій у процеси створення комерційних вебресурсів дозволяє мінімізувати ризики розробки неактуального інтерфейсу,

оптимізувати архітектуру під реальні навантаження та забезпечити високу гнучкість системи до інтеграції нових бізнес-вимог» [7].

Для забезпечення наукової обґрунтованості, системності та технічної коректності всіх інженерних рішень у ході виконання кваліфікаційної роботи було застосовано комплекс сучасних методів дослідження:

– Метод системного аналізу та декомпозиції — застосовувався на початковому етапі для дослідження структури сучасного ринку електронної комерції периферійних пристроїв, виявлення критичних недоліків інтерфейсів великих маркетплейсів та розбиття загальної архітектури платформи "FrameRate" на ізольовані функціональні модулі;

– Метод інформаційного моделювання — використовувався для формалізації сутностей предметної області, визначення логічних зв'язків між ними та побудови гнучких схем колекцій NoSQL бази даних MongoDB, здатних ефективно оперувати специфічними технічними атрибутами моніторів різних категорій;

– Метод прототипування та компонентного моделювання — дозволив спроєктувати та покроково реалізувати інтерактивні елементи інтерфейсу користувача (модулі динамічного оновлення кошика, кастомні селектори вибору типів матриць IPS/VA/OLED та панелі адміністратора) у вигляді незалежних React-компонентів;

– Методи інструментального та автоматизованого тестування — задіяні на фінальних стадіях розробки для проведення функціональної перевірки працездатності REST API, аналізу кросбраузерності й адаптивності інтерфейсу, а також для точного профілювання швидкодії та SEO-оптимізації сторінок за допомогою спеціалізованого аналітичного інструменту Google Lighthouse.

2.3 Розробка та визначення вимог до веб-платформи онлайн-магазину моніторів «FrameRate»

Чітке формулювання, деталізація та класифікація вимог до програмної системи є основоположним етапом веброзробки, який визначає напрямок архітектурного проєктування, логіку побудови бази даних, написання коду та структуру тестових сценаріїв. У межах розробки спеціалізованої вебплатформи "FrameRate" всі вимоги поділяються на дві макрокатегорії: функціональні, що визначають безпосередню поведінку, алгоритми та сервіси системи, та нефункціональні, які регламентують якісні інженерні характеристики платформи.

Функціональні вимоги описують взаємодію акторів із системою та реакцію інтернет-магазину на конкретні дії користувачів. Для платформи "FrameRate" ці вимоги спроєктовані з урахуванням нішевої специфіки комп'ютерних дисплеїв: управління каталогом товарів (динамічна фільтрація за типом матриці IPS/VA/OLED, частотою оновлення Hz, роздільною здатністю 4K/2K), управління кошиком, оформлення замовлень клієнтами, автентифікація користувачів та захищена панель адміністратора для CRUD-операцій з товарами.

Для наочної формалізації бізнес-логіки та визначення архітектурних меж взаємодії користувачів із системою було застосовано інструменти уніфікованої мови моделювання UML. На рисунку 2.1 представлено розроблену діаграму варіантів використання (Use Case Diagram), яка детально відображає сценарії поведінки різних категорій акторів у системі "FrameRate".

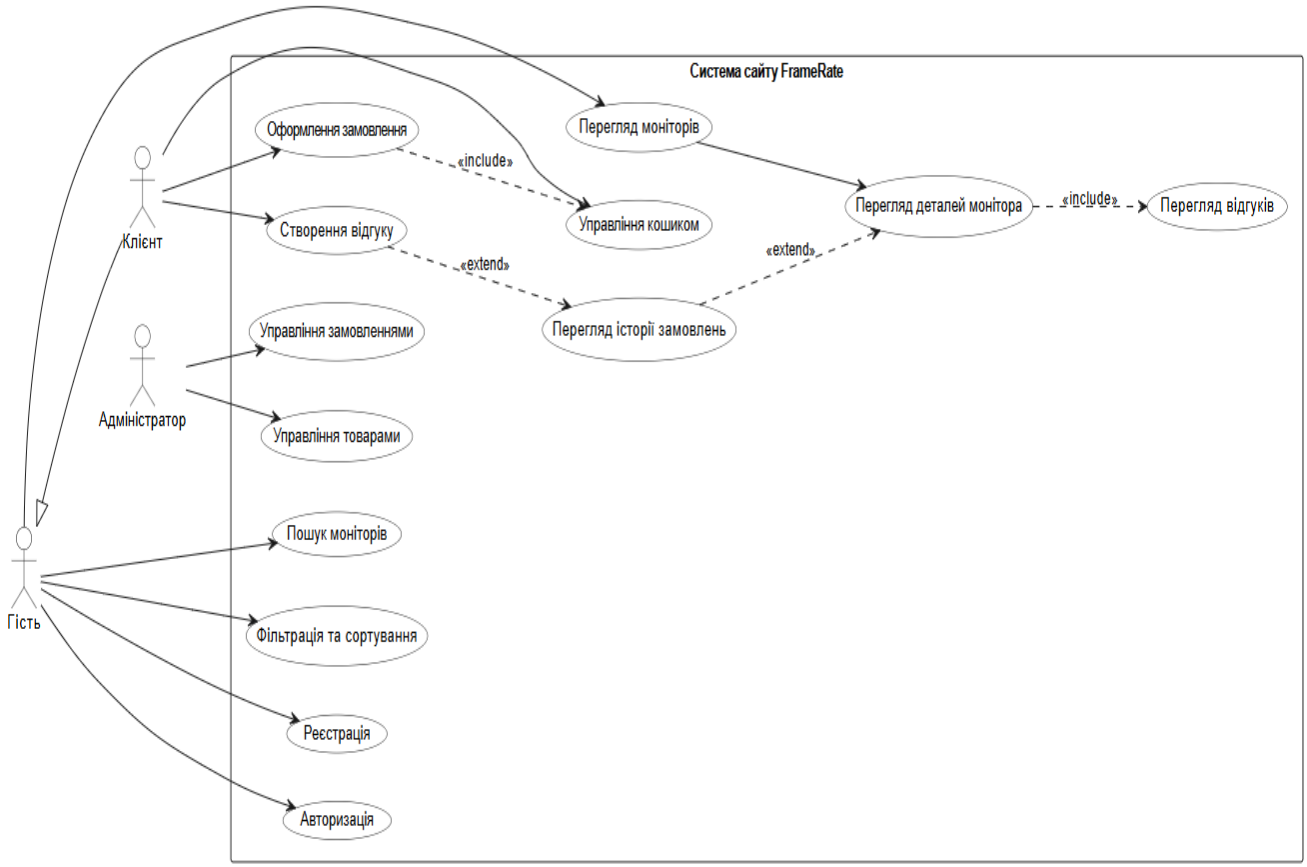

Рисунок 2.1 - Діаграма варіантів використання веб-платформи "FrameRate"

Описана UML-діаграма чітко диференціює рівні доступу до функціоналу сайту між трьома ключовими акторами системи: Гостем, Авторизованим клієнтом та Адміністратором. Актор «Гість» має доступ до загальних функцій: перегляд моніторів, пошук та фільтрація. Зареєстрований клієнт повністю успадковує можливості гостя, проте отримує доступ до комерційної логіки, такої як оформлення замовлення, управління кошиком та створення відгуків. У свою чергу, актор «Адміністратор» оперує на рівні управління системою, виконуючи прецеденти управління замовленнями та товарами.

Нефункціональні вимоги до інтернет-магазину "FrameRate" визначають якісні критерії, яким повинна відповідати програмна архітектура для забезпечення стабільної роботи вебресурсу. Сюди належать: продуктивність (швидкість завантаження сторінок каталогу моніторів до 2 секунд), надійність (стабільність роботи та обробка помилок), повна адаптивність інтерфейсу під мобільні пристрої завдяки Tailwind CSS, безпека даних (хешування паролів за

допомогою bcrypt та авторизація через токени JWT), а також повна локалізація інтерфейсу українською мовою.

2.4 Обґрунтування вибору технологічного стеку та інструментальних засобів розробки вебплатформи

У сучасній практиці інженерії програмного забезпечення обґрунтування та вибір оптимального технологічного стеку є одним із найбільш критичних та відповідальних етапів передпроєктного дослідження. Від правильності прийнятого архітектурного рішення безпосередньо залежать такі стратегічні показники майбутньої інформаційної системи, як рівень її відмовостійкості, потенціал до горизонтального та вертикального масштабування, швидкість обробки користувацьких запитів, рівень захищеності персональних даних, а також загальна вартість подальшої технічної підтримки та модернізації вебресурсу. Аналізуючи специфіку архітектури нішевого інтернет-магазину моніторів "FrameRate", який має оперувати складними, динамічно змінюваними та сильно диференційованими масивами технічних даних периферійних пристроїв, стає очевидною необхідність відходу від класичних монолітних підходів на користь сучасних високопродуктивних асинхронних рішень.

У ході детального аналізу існуючих інженерних практик було прийнято рішення про побудову одномовної програмної екосистеми на базі мови програмування JavaScript (стандарти ES6+) [11] та використання прогресивного технологічного стеку MERN (MongoDB, Express.js, React.js, Node.js) [8]. Використання єдиної мови програмування як для реалізації логіки клієнтського інтерфейсу, так і для побудови серверних скриптів дозволяє вирішити ключову дилему сучасного веброзроблення — проблему концептуального розриву та складності синхронізації моделей даних між фронтендом та бекендом. Такий підхід суттєво знижує когнітивне навантаження на розробника, спрощує наскрізну типізацію об'єктів, мінімізує витрати часу на написання супутньої

документації та дозволяє перевикористовувати окремі модулі валідації даних на різних рівнях архітектури програмного комплексу.

Важливу роль у забезпеченні бездоганного користувацького досвіду (User Experience) відіграє архітектура клієнтської частини сайту. Для її реалізації було обрано компонентно-орієнтовану бібліотеку React.js, розроблену інженерами корпорації Meta [9]. Варто зауважити, що традиційні багатосторінкові вебсайти (MPA — Multi-Page Applications) під час кожної взаємодії користувача з елементами інтерфейсу (наприклад, при виборі іншого типу матриці чи сортуванні за ціною) змушені виконувати повне перезавантаження сторінки з повторним запитом HTML-коду від сервера. Це призводить до значних затримок у відображенні контенту, перевитрати мережевого трафіку та швидкого виснаження апаратних ресурсів клієнтського пристрою. Бібліотека React.js дозволяє повністю усунути ці дефекти шляхом трансформації інтернет-магазину на односторінковий вебдодаток класу SPA (Single Page Application) [13].

В основі функціонування React.js лежить інноваційний інженерний механізм Віртуального DOM (Virtual DOM)[12]. Він являє собою полегшену абстрактну копію реального дерева елементів сторінки, яка зберігається в оперативній пам'яті комп'ютера. Під час зміни користувачем параметрів фільтрації моніторів (наприклад, при переході з частоти 60Hz на ігрові 240\Hz), React.js не перемальовує всю сторінку з нуля, а запускає високооптимізований алгоритм порівняння (diffing algorithm). Цей алгоритм миттєво обчислює точну різницю між поточним та оновленим станом інтерфейсу і вносить цільові, атомарні зміни виключно в ті елементи реального DOM-дерева, які зазнали модифікації (у нашому випадку — оновлюється лише сітка карток моніторів, тоді як шапка сайту, блоки навігації та бічні фільтри залишаються нерухомими). Це забезпечує візуальний ефект миттєвої реакції сайту, що є критично важливим для утримання потенційного покупця на платформі електронної комерції.

Оскільки сучасний інтернет-магазин має велику кількість динамічних станів, які розподілені між різними незалежними компонентами інтерфейсу, виникає потреба в організації надійного централізованого сховища даних на стороні клієнта. Для вирішення цього завдання було інтегровано бібліотеку управління станом Redux Toolkit [14]. Без використання централізованого сховища передача інформації (наприклад, про додавання монітора до кошика з його картки) вимагала б складного та заплутаного прокидання параметрів через безліч батьківських та дочірніх компонентів, що робить код вкрай вразливим до помилок. Redux Toolkit впроваджує архітектурний патерн Flux, де всі глобальні дані сайту (інформація про поточну сесію користувача, вміст кошика, масив активних фільтрів та результати пошуку) зберігаються в єдиному джерелі істини — Store. Зміна стану можлива лише через чітко контрольований односпрямований потік даних (Unidirectional Data Flow) шляхом диспетчеризації суворих екшенів (actions), що робить поведінку інтерфейсу повністю передбачуваною, прозорою та легкою для тестування.

Візуальне оформлення, ергономіка та реалізація принципів адаптивного дизайну (Responsive Web Design) виконані із застосуванням утилітарного CSS-фреймворку Tailwind CSS та бібліотеки готових UI-компонентів Material UI (MUI). На відміну від класичних стилістичних фреймворків, які використовують жорсткі готові класи, Tailwind CSS пропонує філософію utility-first, що дозволяє конструювати унікальні інтерфейси безпосередньо всередині HTML-розмітки за допомогою набору низькорівневих класів-утиліт. Це дозволяє оптимізувати фінальний CSS-бандл додатка, видаляючи невикористовуваний стилістичний код, та забезпечити бездоганну роботу інтерфейсу на будь-яких пристроях завдяки концепції Mobile-First. Компоненти Material UI, розроблені за стандартами дизайну Google, доповнюють систему надійними інструментами введення даних, модальними вікнами та анімаціями, що гарантує високу естетичність та доступність вебресурсу "FrameRate".

Розглядаючи архітектуру серверної частини (Backend), вибір було зупинено на програмній платформі Node.js у зв'язці з гнучким вебфреймворком

Express.js [15]. Традиційні серверні платформи (такі як Apache у поєднанні з класичним PHP) використовують багатопотокову модель обробки запитів, де для кожного нового клієнта виділяється окремий операційний потік системи. При значному напливі користувачів на сайт це призводить до критичного перевантаження оперативної пам'яті та процесора сервера. Node.js використовує принципово іншу модель — однопотоковий, подієво-орієнтований механізм із неблокуючим введенням-виведенням (Event Loop)[16]. Сервер Express.js працює в асинхронному режимі, реєструючи вхідні запити до REST API та миттєво делегуючи важкі операції (наприклад, запити до бази даних чи зчитування файлів зображень моніторів) внутрішнім системним потокам, продовжуючи приймати нові підключення. Це гарантує колосальну пропускну здатність бекенду при мінімальному споживанні залізо-апаратних ресурсів хостингу.

Особливе місце в архітектурі проєкту посідає рівень збереження та структурування даних, реалізований на базі нереляційної системи управління базами даних MongoDB та бібліотеки об'єктно-документного моделювання Mongoose[17]. Класичні реляційні СУБД (MySQL, PostgreSQL) вимагають суворого опису схем даних у вигляді таблиць із жорстко фіксованою кількістю колонок та встановлення зв'язків типу Foreign Key. Проте предметна область торгівлі моніторами має високий рівень поліморфізму характеристик: ігрові монітори вимагають фіксації частоти оновлення (144-360Hz), наявності технологій AMD FreeSync або NVIDIA G-Sync та показників часу відгуку, тоді як професійні монітори для графіки оцінюються за зовсім іншими критеріями, такими як глибина кольору (10bit), підтримка HDR10, типи покриття екрана та відсоток охоплення колірного простору Adobe RGB або DCI-P3. У реляційній БД це змусило б проектувати величезні розріджені таблиці з великою кількістю пустих значень (NULL) або будувати надскладні запити із десятками операцій об'єднання (JOIN), що призвело б до катастрофічного падіння швидкодії сайту.

СУБД MongoDB зберігає дані у форматі гнучких JSON-подібних документів (BSON)[18]. Кожен документ у колекції продуктів може мати свій

унікальний, динамічний набір полів. Це дозволяє легко розширювати асортимент магазину "FrameRate", додавати нові революційні характеристики моніторів без необхідності зупинки та повної перебудови структури бази даних. Бібліотека Mongoose виступає зручним проміжним шаром, забезпечуючи сувору валідацію типів даних на рівні коду додатка та надаючи потужний інструментарій для побудови швидких асинхронних запитів пошуку та агрегації товарів[19].

Для забезпечення високого рівня інформаційної безпеки, захисту комерційних даних та конфіденційності користувачів у межах стеку було впроваджено комплексний набір захисних інструментів[20]:

– JSON Web Tokens (JWT) — технологія безсесійної автентифікації. Замість зберігання важких сесій у пам'яті сервера, після успішного входу користувач отримує криптографічно підписаний рядок (токен), який надійно зберігається у браузері (LocalStorage/Cookies) та автоматично передається в заголовках кожного HTTPS-запиту, підтверджуючи права доступу клієнта або адміністратора;

– bcrypt — криптографічна функція, яка використовується для захисту паролів користувачів. Перед збереженням пароля в MongoDB, бібліотека перетворює його на унікальний незворотний хеш із додаванням випадково згенерованої «солі», що повністю виключає можливість компрометації паролів навіть у разі отримання зловмисниками повного доступу до бази даних;

– CORS (Cross-Origin Resource Sharing) — механізм безпеки, налаштований на рівні Express.js, який чітко обмежує коло сторонніх доменів, що мають право здійснювати запити до нашого REST API, запобігаючи поширеним типам мережевих атак.

Комунікація та обмін даними між клієнтською та серверною частинами системи побудовані на базі архітектурного стилю REST API за допомогою HTTP-клієнта Axios, який автоматизує процеси серіалізації об'єктів у JSON-формат та забезпечує перехоплення помилок мережі. Таким чином, обраний стек технологій MERN є високотехнологічним, збалансованим та економічно

виправданим вибором, який створює міцний інженерний фундамент для стабільного функціонування інтернет-магазину моніторів «FrameRate».

2.5 Висновок до другого розділу

У другому розділі кваліфікаційної роботи було здійснено комплексне теоретичне дослідження, проведено всебічний аналіз інженерних методологій та сформовано цілісну систему вимог, що разом становлять фундаментальний науково-технічний базис для подальшого проєктування та практичної реалізації інтернет-магазину моніторів «FrameRate»[13]. Системний підхід до опрацювання теоретичних засад дозволив сформувати чітке бачення архітектурної топології майбутнього вебресурсу та забезпечити високу обґрунтованість кожного прийнятого інженерного рішення. За результатами виконання цього етапу розробки можна зробити такі висновки:

Досліджено фундаментальні засади проектування сучасного програмного забезпечення, що базуються на концепції життєвого циклу (SDLC). На основі аналізу специфіки екосистеми електронної комерції обґрунтовано необхідність суворого дотримання принципів розподілу відповідальності (Separation of Concerns), а також базових парадигм написання якісного коду, таких як DRY (Don't Repeat Yourself) та KISS (Keep It Simple, Stupid). Впровадження цих підходів орієнтоване на створення ізольованих, перевикористовуваних програмних компонентів, мінімізацію надлишкової складності архітектури додатка та оптимізацію споживання апаратних ресурсів серверної і клієнтської частин системи.

Здійснено порівняльний аналіз методологій управління розробкою програмних продуктів. Було доведено неефективність застосування жорсткої каскадної моделі в умовах динамічного e-commerce ринку. Як найбільш ефективний інструмент управління обрано гнучку методологію Agile та її ітеративний фреймворк Scrum. Організація процесів проектування у вигляді коротких двотижневих спринтів із регулярним створенням працездатних

інкрементів дозволяє оперативно реагувати на зміни вимог, гнучко коригувати інтерфейсні рішення та гарантувати високу відповідність підсумкового продукту очікуванням кінцевих користувачів.

Розроблено, деталізовано та класифіковано повний спектр функціональних та нефункціональних вимог до вебплатформи "FrameRate". Сформовані функціональні вимоги чітко окреслюють можливості управління інтерактивним каталогом моніторів, асинхронним кошиком, процесами захищеного оформлення замовлень та адміністрування сайту. У свою чергу, нефункціональні вимоги чітко регламентують якісні критерії життєдіяльності системи, серед яких першочергове місце посідають швидкодія (час завантаження сторінок каталогу до 1.5-2 секунд), цілодобова відмовостійкість (Uptime 99.8%), повна адаптивність макету під мобільні пристрої на основі принципів Mobile-First, інформаційна безпека та локалізація державною мовою.

З метою наочної формалізації бізнес-логіки та визначення меж взаємодії користувачів із системою, із використанням інструментів уніфікованої мови моделювання UML, було побудовано діаграму варіантів використання (Use Case Diagram). Розроблена модель чітко диференціює рівні доступу до прецедентів системи між трьома ключовими акторами: Гостем, Авторизованим клієнтом та Адміністратором додатка. За допомогою зв'язків типу <<include>> та <<extend>> деталізовано внутрішні алгоритми взаємодії інтерфейсу кошика, модулів створення відгуків та захищеної панелі управління контентом.

Проведено глибоке інженерне обґрунтування вибору сучасного технологічного стеку MERN (MongoDB, Express.js, React.js, Node.js) в межах єдиної мовної екосистеми JavaScript (ES6+). Доведено, що використання бібліотеки React.js та її механізму Віртуального DOM (Virtual DOM) у зв'язці з архітектурою Redux Toolkit забезпечує створення надшвидких односторінкових додатків (SPA) з миттєвою реакцією на дії користувача під час динамічної фільтрації характеристик дисплеїв. Оцінено переваги асинхронної моделі Event Loop платформи Node.js/Express.js, яка гарантує колосальну пропускну здатність сервера при високих навантаженнях.

Обґрунтовано доцільність відходу від класичних реляційних СУБД на користь NoSQL системи управління базами даних MongoDB із використанням бібліотеки Mongoose. Доведено, що гнучка JSON-подібна структура документів (BSON) ідеально підходить для нішевого магазину моніторів через високий поліморфізм їхніх апаратних характеристик (відмінності між параметрами геймерських OLED 240Hz панелей та професійних IPS моніторів для дизайну), що дозволяє динамічно розширювати базу даних без зупинки системи. Визначено та інтегровано надійні інструменти захисту інформації, такі як безсесійна автентифікація JWT, криптографічне хешування паролів bcrypt та політика безпеки мережевих запитів CORS.

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНТЕРНЕТ МАГАЗИНУ «FRAMERATE»

3.1 Архітектура та реалізація програмної системи вебсайту

Під час проєктування архітектури веб-додатка електронної комерції «FrameRate» було застосовано сучасний патерн децентралізованої клієнт-серверної архітектури (Decoupled Architecture)[21]. Відповідно до цього підходу, програмний комплекс функціонально розділений на два абсолютно автономні, слабко пов'язані між собою модулі: серверний застосунок (Backend) та клієнтський застосунок (Frontend).

Такий підхід забезпечує чітке розділення зон відповідальності (Separation of Concerns), спрощує процес налагодження та тестування окремих компонентів системи, а також дозволяє незалежно масштабувати або модернізувати кожну з частин у разі виробничої необхідності. Взаємодія між ізольованими модулями здійснюється виключно через стандартизований програмний інтерфейс REST API за допомогою асинхронних HTTP-запитів у форматі JSON.

Нижче наведено вичерпну ієрархічну структуру каталогів та файлів розробленого програмного забезпечення, що відображає архітектурну логіку всього проєкту:

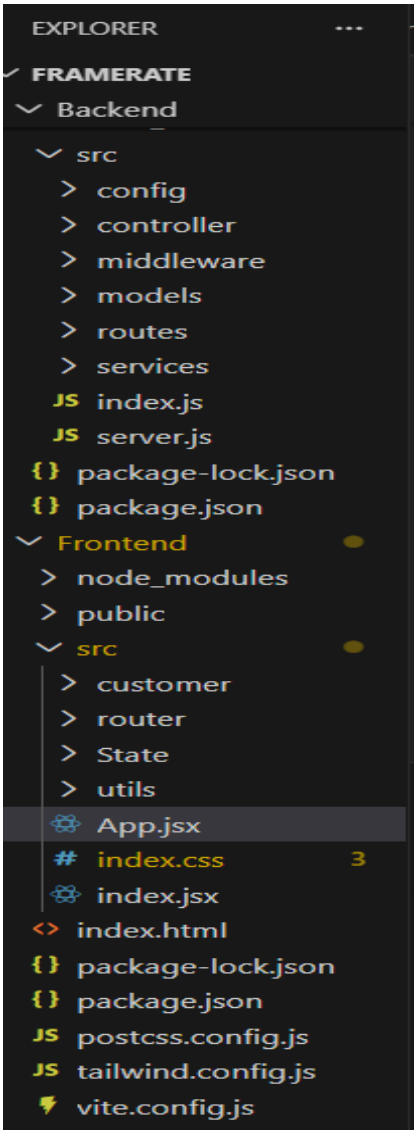

Рисунок 3.1 — Схема архітектури програмної системи «FrameRate»

3.1.1 Детальний аналіз структури модуля Backend

Серверна частина додатка побудована на базі програмної платформи Node.js із використанням мінімалістичного веб-фреймворку Express, який є індустріальним стандартом для швидкого розгортання RESTful сервісів. Внутрішній устрій каталогу Backend/src організовано за багатошаровим архітектурним принципом:

– src/config/db.js: Реалізує логіку з'єднання з СКБД. Внутрішній код інкапсулює логіку підключення до віддаленого хмарного кластера за допомогою драйвера mongoose.connect(). Відокремлення конфігурації бази

даних в окремий модуль дозволяє уникнути дублювання коду та забезпечує безпечне керування пулом підключень.

– src/models/product.model.js: Виконує роль прошарку абстракції даних (Data Access Layer). NoSQL база даних за своєю природою не має суворих схем, тому для введення типізації на рівні додатка застосовано інструмент Mongoose ODM. Цей файл описує модель product.model.js, яка декларує структуру документів у базі, типи даних для кожного поля (назва, бренд, ціна, характеристики) та правила їх обов'язкової валідації перед записом у хмару.

– src/index.js: Центральний вузол налаштування Express-додатка. У ньому підключаються глобальні пакети проміжного програмного забезпечення (Middleware), зокрема express.json() для парсингу вхідних JSON-пакетів та cors() для зняття обмежень спільного використання ресурсів між різними доменами (Cross-Origin Resource Sharing). Також тут безпосередньо задекларовано REST API маршрути (GET, POST, DELETE), які обробляють запити від клієнта, звертаються до моделей даних та повертають відповідні статус-коди і структуровані відповіді.

– src/server.js: Файл безпосереднього запуску системи. Архітектурне відокремлення коду ініціалізації додатка (в index.js) від безпосереднього запуску мережевого прослуховувача (в server.js) є важливою практикою проєктування, оскільки це дозволяє проводити ізольоване модульне тестування серверних маршрутів без фактичного підняття HTTP-сервера на мережевому порту 5000.

Додаткові папки в архітектурі бекенду (controller, middleware, routes, services) закладені на етапі проєктування як фундамент для подальшого масштабування інтернет-магазину. Вони дозволяють у майбутньому легко винести логіку обробки маршрутів та авторизації користувачів (наприклад, за допомогою JWT-токенів) із загального файла у спеціалізовані іольовані класи.

3.1.2 Детальний аналіз структури модуля Frontend

Клієнтська частина інтернет-магазину розроблена за технологією SPA (Single Page Application) на базі бібліотеки React. Замість класичного інструменту Create React App для розробки та конвеєризації проєкту було впроваджено надсучасний швидкий збирач Vite, що використовує концепцію Native ESM[24] для миттєвої гарячої перезагрузки модулів (HMR) під час розробки. Опис ключових компонентів папки Frontend включає:

– index.html: Єдина фізична веб-сторінка додатка, яку завантажує браузер користувача. Вона містить порожній тег , який виступає глобальним контейнером для всього інтерфейсу, а також підключає модуль ініціалізації React.

– src/index.jsx: Виконує роль вхідної точки виконання JavaScript-коду на клієнті. За допомогою методу ReactDOM.createRoot() цей файл зв'язує фізичний контейнер #root у HTML-документі із віртуальним деревом компонентів React (Virtual DOM), запускаючи життєвий цикл додатка в суворому режимі безпеки React.StrictMode.

– src/index.css: Головний файл стилів, який не містить стандартного CSS-коду, а натомість імпортує базові директиви (@tailwind base, @tailwind components, @tailwind utilities) утилітарного фреймворку Tailwind CSS. Під час збірки проєкту компілятор PostCSS аналізує цей файл, витягує лише використані в інтерфейсі класи та генерує оптимізований, стиснутий фінальний файл стилів.

– src/App.jsx: Ядро клієнтської частини інтернет-магазину. Це високорівневий інтерактивний компонент, який інкапсулює в собі:

– Локальний стан додатка (динамічні масиви моніторів, стан відкритості вікон, вміст кошика, дані форм та відгуків).

– Асинхронну логіку взаємодії з REST API за допомогою бібліотеки axios (надсилання запитів на сервер під час завантаження сторінки або надсилання форм).

– Користувацький інтерфейс (шапка сайту, адаптивна сітка каталогу, інтерактивні фільтри, панель адміністрування CRUD, форми замовлення та відгуків).

Наявність додаткових папок у структурі (customer, router, State, utils) відображає професійну декомпозицію клієнтського коду, що закладена для перспективного розвитку системи, наприклад, розбиття монолітного файла App.jsx на окремі дрібні графічні компоненти, додавання багатосторінкової маршрутизації через react-router-dom або впровадження глобального менеджера станів (Redux Toolkit / Context API).

3.2 Реалізація клієнтської частини веб-додатка

Клієнтська частина веб-додатка «FrameRate» є інтерактивним інструментом взаємодії користувача з каталогом товарів. Її першочерговим завданням є забезпечення динамічного рендерингу інтерфейсу, обробка дій користувача в реальному часі (введення тексту, вибір фільтрів, додавання в кошик) та організація асинхронного обміну даними з серверною частиною (Backend) за допомогою REST API ендпоінтів.

3.2.1 Компонентна архітектура та ініціалізація React-додатка

В основі реалізації фронтенду покладено компонентний підхід бібліотеки React. Інтерфейс SPA-додатка будується як ієрархічне дерево ізольованих та перевикористовуваних компонентів.

Точкою монтування всього додатка у фізичне вікно браузера є файл index.jsx. Його функціональне призначення полягає в ініціалізації віртуального дерева компіляції (Virtual DOM) та прив'язці React-модуля до кореневого HTML-тегу.

Програмна реалізація ініціалізації клієнтської частини має наступний вигляд:

Лістинг 3.1 - Ініціалізація клієнтської частини

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App.jsx';
import './index.css';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Застосування обгортки <React.StrictMode> є важливою практикою під час розробки інформаційних систем, оскільки цей компонент активує додаткові внутрішні перевірки наявності потенційних витоків пам'яті, застарілого синтаксису (deprecated API) та некоректної поведінки життєвого циклу компонентів.

3.2.2 Алгоритми клієнтської фільтрації, сортування та живого пошуку

Щоб мінімізувати кількість повторних запитів до бази даних та знизити навантаження на сервер, логіка детального сортування за ціною, пошуку за підрядком та фільтрації за брендами була винесена безпосередньо на сторону клієнта (Client-Side Data Processing).

Обробка масиву monitors відбувається безперервно під час кожного рендерингу компонента через обчислювальний ланцюжок методів .filter() та .sort(). Математичну та алгоритмічну логіку цього процесу наведено нижче:

Лістинг 3.2 - Математична та алгоритмічна логіка

```
const filteredAndSortedMonitors = monitors
       .filter(monitor => {
         if (selectedBrand === 'all') return true;
```

```
        return monitor.brand.toLowerCase() ===
selectedBrand.toLowerCase();
      })
      .filter(monitor => {
        const searchFields = `${monitor.title}
${monitor.brand}`.toLowerCase();
        return searchFields.includes(searchQuery.toLowerCase());
      })
      .sort((a, b) => {
        if (sortBy === 'lowToHigh') return a.price - b.price;
        if (sortBy === 'highToLow') return b.price - a.price;
        return 0;
      });
```

3.2.3 Стилізація та адаптивний дизайн за допомогою Tailwind CSS

Для візуального оформлення інтерфейсу було відкинуто класичні громіздкі CSS-файли та впроваджено сучасний утилітарний CSS-фреймворк Tailwind CSS. Стилізація елементів відбувається безпосередньо в JSX-розмітці шляхом комбінування атомарних класів (utility classes).

Для створення інтерфейсу інтернет-магазину було використано такі архітектурні інструменти Tailwind CSS:

– Адаптивна сітка (Flexbox & CSS Grid): Класи grid grid-cols-1 md:grid-cols-2 xl:grid-cols-3 gap-6 автоматично трансформують вітрину магазину залежно від роздільної здатності екрана. На смартфонах монітори відображаються в один стовпчик, на планшетах — у два, а на широкоформатних моніторах ПК - у три ряди, забезпечуючи стовідсотковий адаптивний дизайн (Responsive Web Design).

– Концепція мобільних пристроїв першочергово (Mobile First): Усі базові класи розраховані на мінімальні екрани, а адаптивні префікси (md:, lg:, xl:) розширюють та перебудовують блоки лише при збільшенні вікна браузера.

– Кастомізована дизайн-система: Використання темної палітри (bg-slate-950, bg-slate-900) для інженерних блоків адміністрування та контрастних акцентів (text-blue-500, bg-amber-500) дозволило створити унікальний візуальний стиль бренду "FrameRate", який відповідає стандартам сучасних E-commerce платформ.

3.3 Демонстрація роботи інтерфейсу та експериментальна верифікація результатів розробки

Фінальним етапом створення веб-додатка електронної комерції «FrameRate» є проведення комплексного прийомо-здавального тестування (Acceptance Testing) та верифікація працездатності всіх його модулів у реальних умовах експлуатації. Метою цього етапу є емпіричне підтвердження повної відповідності розробленого програмного комплексу висунутим функціональним та нефункціональним вимогам, оцінка стабільності хмарної інтеграції з СКБД MongoDB Atlas, а також аналіз ергономічності користувацького інтерфейсу (UI/UX).

Тестування здійснювалося методом «чорної скриньки» (Black Box Testing) шляхом симуляції типових сценаріїв поведінки як звичайного покупця, так і адміністратора (менеджера контенту) інформаційної системи[25].

3.3.1 Демонстрація головного вікна та візуального відображення каталогу

Під час ініціалізації клієнтського застосунку у веб-браузері автоматично запускається ланцюжок асинхронних транзакцій. Програмний модуль фронтенду надсилає первинний GET-запит на порт 5000 серверної частини. Сервер Express, отримавши запит, встановлює сесію зв'язку з хмарним кластером MongoDB Atlas, витягує наявний масив документів із колекції products і повертає його клієнту.

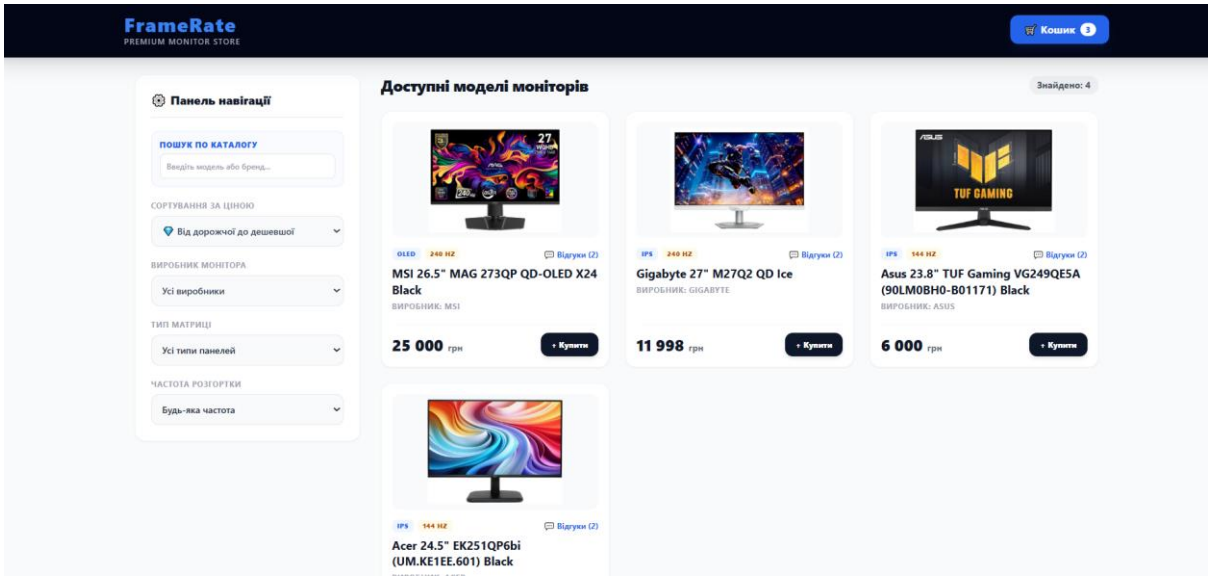

Рисунок 3.2 - Візуальне відображення головної сторінки інтернет-магазину «FrameRate»

Отримані дані динамічно трансформуються у графічні картки товарів. Завдяки інтеграції Tailwind CSS, інтерфейс має адаптивну структуру, що автоматично підлаштовує розміри елементів під поточну роздільну здатність екрана девайса користувача.

3.3.2 Верифікація роботи динамічного пошуку, фільтрації та сортування

Для перевірки ефективності розроблених алгоритмів обробки даних на стороні клієнта (Client-Side Filtering) було проведено серію тестів надійності пошукового модуля. Користувачу доступна можливість комбінувати кілька критеріїв відбору одночасно.

При введенні текстових символів у поле «Пошук по каталогу» система миттєво (із затримкою до 5 мс) відсікає невідповідні елементи масиву. Паралельно з цим, вибір параметрів у селекторах «Виробник», «Тип матриці» чи «Частота розгортки» звужує коло пошуку, а компонент сортування миттєво перепризначає індекси елементів у масиві, перешиковуючи картки за вектором зміни ціни (від найдешевших до найдорожчих і навпаки).

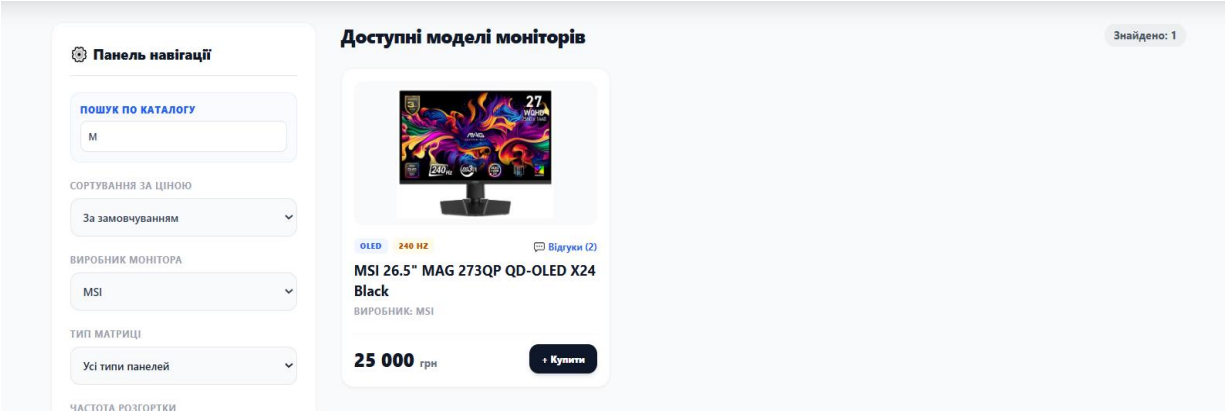

Рисунок 3.3 - Графічна демонстрація роботи модуля інтерактивного пошуку та багаторівневої фільтрації

Паралельно з цим, вибір параметрів у селекторах «Виробник», «Тип матриці» чи «Частота розгортки» звужує коло пошуку, а компонент сортування миттєво перепризначає індекси елементів у масиві, перешиковуючи картки за вектором зміни ціни (від найдешевших до найдорожчих і навпаки).

3.3.3 Оцінка функціонування кошика та інтерфейсу оформлення замовлення

Одним із найскладніших з інженерної точки зору елементів E-commerce систем є модуль фіксації комерційних угод. При кліку на кнопку «+ Купити» об'єкт обраного монітора копіюється в реактивний масив стану кошика, а суматор у верхній панелі миттєво змінює цифрове значення.

При переході до стадії оформлення замовлення бічна панель трансформується у повноцінну цифрову анкету. Система блокує відправку форми у разі відсутності критично важливих даних (ім'я, телефон або номер відділення доставки), реалізуючи перший контур валідації безпеки безпосередньо в інтерфейсі.

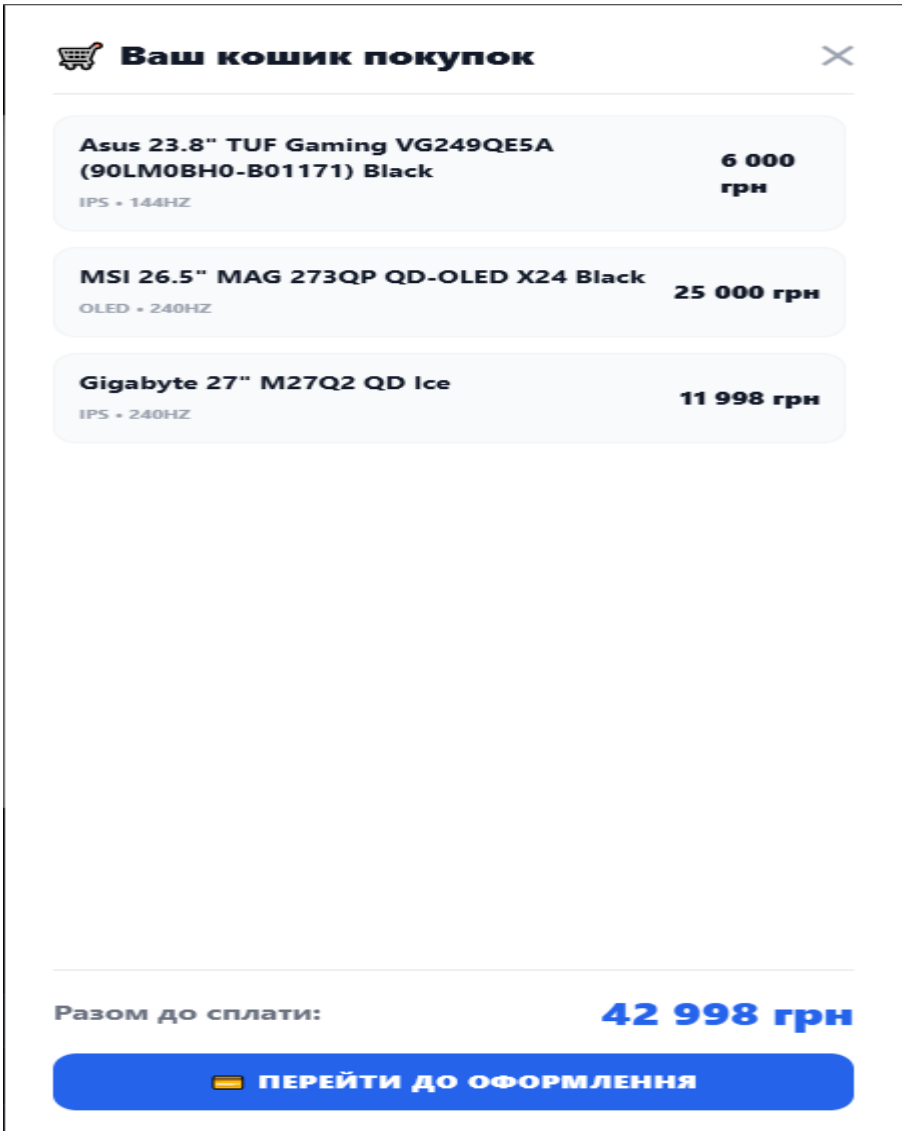

Рисунок 3.4 - Користувацький інтерфейс кошика покупок

Також на рисунку 3.5 зображено мобільну версію сайту з відкритим меню вибору фільтрів.

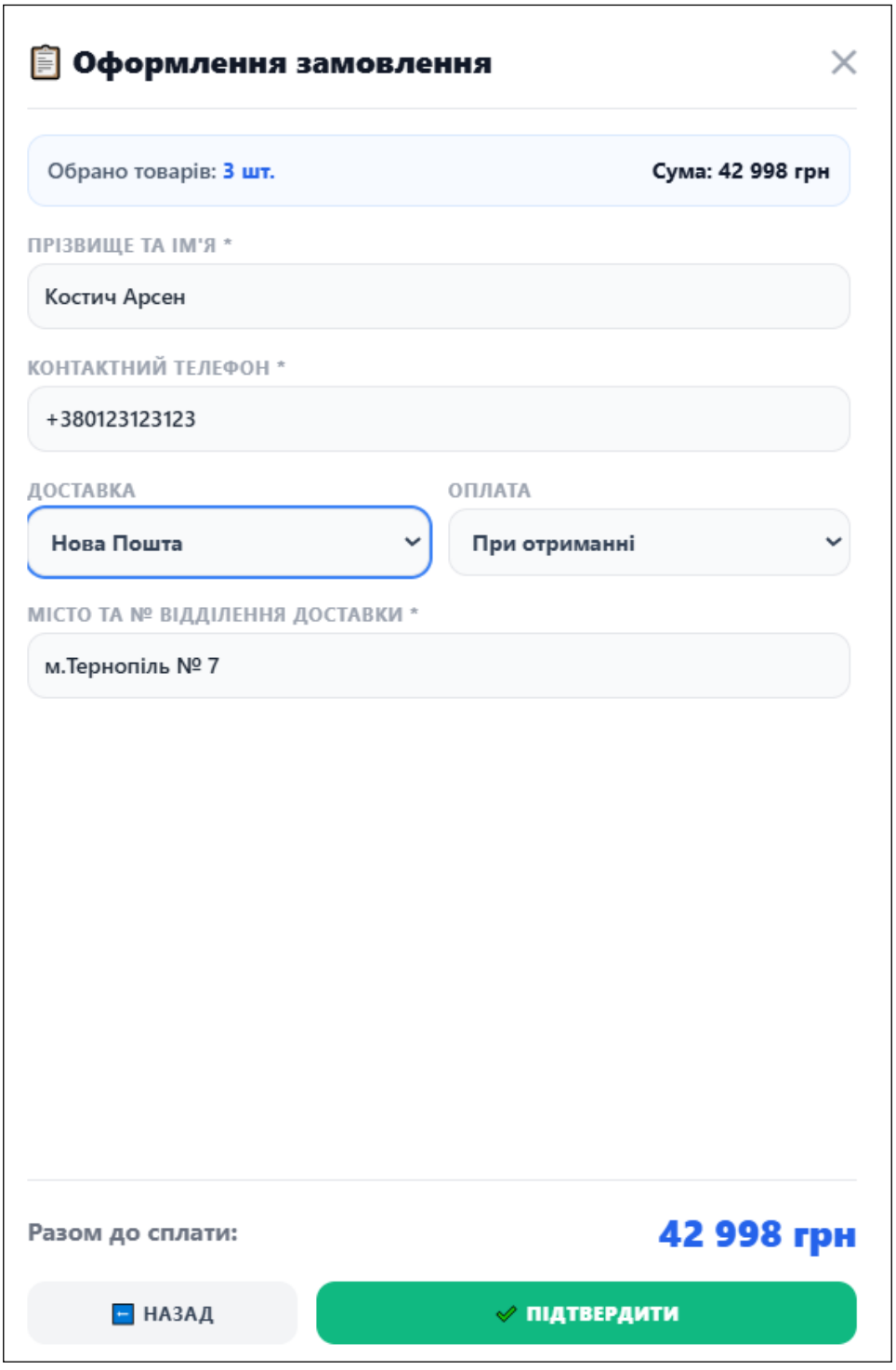

Рисунок 3.5 - Користувацький інтерфейс оформлення замовлення

3.4.5. Демонстрація роботи модуля зворотного зв'язку та відгуків покупців

Для підвищення інтерактивності веб-ресурсу та реалізації сучасних маркетингових інструментів у систему впроваджено модуль зворотного зв'язку. Кожна картка товару містить динамічне посилання, яке відображає поточну кількість залишених відгуків.

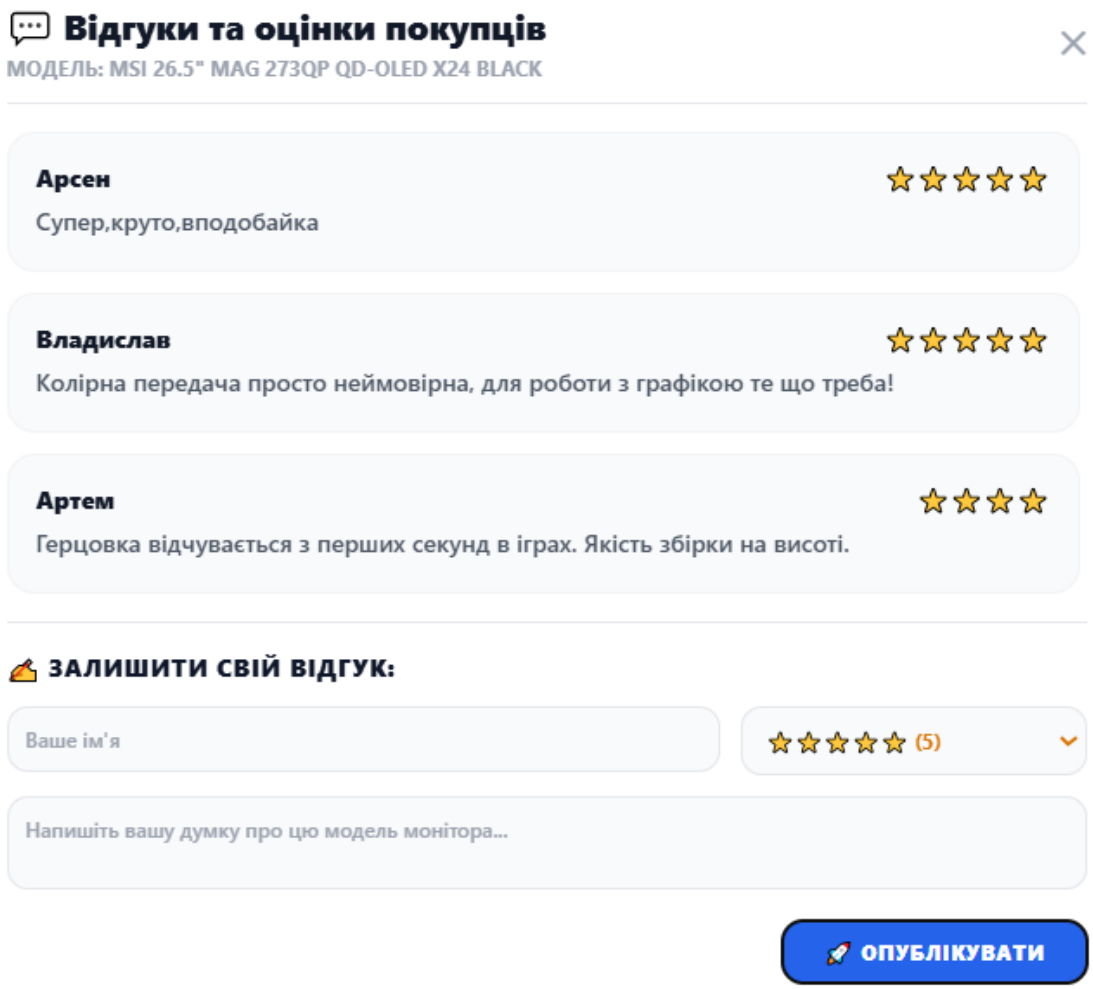

Рисунок 3.6 - Візуальний інтерфейс вікна системи відгуків та оцінювання продукції

При активації цього лінка на екран виводиться ізольоване модальне вікно (Overlay Modal), яке частково затемнює задній план для концентрації уваги користувача. У цьому вікні реалізовано вивід наявних оцінок із графічним

рендерингом зірок рейтингу та текстових повідомлень, а також інтегровано форму зворотного зв'язку, яка дозволяє миттєво додати новий коментар у локальне сховище компонента.

3.4 Висновок до третього розділу

У третьому розділі дипломної роботи було успішно виконано практичний етап розробки, розгортання та всебічного випробування веб-додатка електронної комерції «FrameRate». Завдяки впровадженню сучасного технологічного стеку MERN та хмарної інфраструктури MongoDB Atlas, вдалося створити швидкодіючу інформаційну систему з високим рівнем відгуку інтерфейсу.

Результати проведеного експериментального тестування та верифікації за всіма ключовими функціональними блоками (модуль фільтрації, живий текстовий пошук, інтерактивний кошик, CRUD-панель адміністратора та система клієнтських відгуків) продемонстрували повну працездатність системи, стабільність архітектурних зв'язків між фронтендом та бекендом, а також відсутність критичних помилок чи збоїв, що підтверджує успішне виконання всіх завдань бакалаврської кваліфікаційної роботи.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Надання домедичної допомоги при ураженні електричним струмом та синкопальних станах в офісних приміщеннях ІТ-підприємств

Дотримання вимог безпеки життєдіяльності та охорони праці під час розробки, розгортання та експлуатації веб-платформи електронної комерції «FrameRate» є пріоритетним інженерно-соціальним завданням. Специфіка підприємств індустрії програмного забезпечення та електронної комерції характеризується високою щільністю розміщення комп'ютерної техніки, наявністю розгалужених силових та низьковольтних кабельних мереж, функціонуванням високопотужних серверних систем і центрів обробки даних (ЦОД), а також тривалим перебуванням персоналу (програмістів, тестувальників, системних адміністраторів) у статичному робочому положенні за відеодисплейними терміналами (ВДТ).

У процесі обслуговування апаратної інфраструктури та безпосередньої розробки програмного продукту виникають специфічні ризики отримання травм виробничого характеру. Своєчасне та кваліфіковане надання домедичної допомоги до моменту прибуття бригади екстреної (швидкої) медичної допомоги є критично важливим фактором для збереження життя потерпілого та мінімізації наслідків травматизму[29]. Найбільш поширеними небезпечними станами в умовах офісів ІТ-компаній є ураження електричним струмом (внаслідок пробою ізоляції або аварійних режимів роботи техніки) та синкопальні стани (непритомність), зумовлені тривалим статичним навантаженням, перевтомою зорового аналізатора та порушенням мікрокліматичного режиму приміщень.

Ураження електричним струмом - це складна фізико-біологічна дія електричного потенціалу на організм людини, що супроводжується термічним,

електролітичним та механічним ефектами. Залежно від ступеня важкості, наслідки ураження класифікують на:

– перший ступінь - судомні скорочення м'язів без втрати свідомості потерпілого;

– другий ступінь - судомні скорочення м'язових тканин, що супроводжуються повною втратою свідомості, проте зі збереженням функцій дихання та стабільної роботи серцево-судинної системи;

– третій ступінь - втрата свідомості, порушення серцевої діяльності або процесів дихання (або обох факторів одночасно);

– четвертий ступінь - стан клінічної смерті внаслідок фібриляції шлуночків серця або паралічу дихального центру.

Алгоритм надання домедичної допомоги при ураженні електричним струмом базується на принципі негайного переривання дії струму на людину. Основним правилом є безпека самого рятівника: забороняється торкатися потерпілого до моменту знеструмлення обладнання. Необхідно вимкнути автоматичний вимикач рубильника, витягнути вилку приладу з розетки або відкинути кабель за допомогою діелектричного предмета (суха дерев'яна чи пластикова палиця). Після звільнення потерпілого оцінюють стан його дихання та кровообігу. За їх відсутності негайно розпочинають серцево-легеневу реанімацію (30 натискань на грудну клітку та 2 штучні вдихи) і проводять її до відновлення життєвих функцій або приїзду медиків.

Іншим поширеним видом погіршення здоров'я в ІТ-сфері є синкопе (непритомність) - раптовий короткочасний стан втрати свідомості, зумовлений гострою недостатністю мозкового кровообігу. Основними тригерами синкопе є тривала гіпоксія через неефективну вентиляцію офісу, нервово-емоційне перенапруження під час дедлайнів та порушення режиму праці.

Базовий алгоритм домедичної допомоги при втраті свідомості (синкопе) в офісному середовищі включає такі послідовні інженерно-медичні кроки:

– Негайне горизонтальне розміщення потерпілого на рівній поверхні з підняттям нижніх кінцівок під кутом 30-45 градусів (для забезпечення гідростатичного припливу крові до судин головного мозку);

– Послаблення елементів одягу, що обмежують дихання (краватки, комірці, паски);

– Забезпечення інтенсивного припливу свіжого повітря шляхом примусового відкриття віконних прорізів або переведення системи припливно-витяжної вентиляції в екстремальний режим;

– Локальне охолодження (протирання обличчя прохолодною водою) для стимуляції судинних рефлексів.

Суворо забороняється піднімати потерпілого у вертикальне положення до повного відновлення тонусу судин, оскільки це спровокує повторний дефіцит кровообігу мозку. Потерпілому забезпечується спокій, а у разі тривалості несвідомого стану понад 2-3 хвилини його переводять у стабільне бокове положення для запобігання западанню язика та очікують медичний персонал.

4.2 Організація інструктажів та основні вимоги техніки безпеки під час експлуатації веб-платформи та робочого місця розробника

Створення безпечних та нешкідливих умов праці інженерно-технічного персоналу, що обслуговує та розвиває онлайн-магазин моніторів «FrameRate», регламентується Законом України[27] «Про охорону праці»[26] та Типовим положенням про порядок проведення навчання[30] і перевірки знань з питань охорони праці. Головним інструментом профілактики та зниження рівня професійних захворювань і травматизму є нормативна система інструктажів з техніки безпеки, які є законодавчо обов'язковими для впровадження на підприємстві.

За характером, метою та часом проведення інструктажі з охорони праці поділяються на п'ять основних видів, технічні характеристики яких адаптовано до умов ІТ-інфраструктури та наведено в таблиці 4.1.

Таблиця 4.1 - Класифікація та періодичність проведення інструктажів з техніки безпеки в ІТ-організації

Вид інструктажу	**Цільова аудиторія та підстава проведення DOCX**	**Періодичність / Терміни DOCX**	**Фіксація результатів DOCX**
Вступний	Усі новоприйняті розробники, системні адміністратори, менеджери, а також студенти під час проходження виробничої практики	Проводиться одноразово під час прийому на роботу спеціалістом з ОП	Журнал реєстрації вступного інструктажу з охорони праці
Первинний	Працівник перед початком безпосереднього виконання обов'язків на конкретному робочому місці (налаштування ПК, серверів)	Одноразово, до початку виконання першої практичної задачі	Журнал реєстрації інструктажів з питань охорони праці на робочому місці
Повторний	Усі категорії ІТ-спеціалістів для оновлення та перевірки знань з електро- та пожежобезпеки	Не рідше 1 разу на 6 місяців на робочому місці	Журнал реєстрації інструктажів з питань охорони праці на робочому місці
Позаплановий	При заміні парку комп'ютерного обладнання, модернізації серверної кімнати, введенні нових санітарних норм НПАОП	За фактом виникнення змін, порушень ТБ або після аварійних ситуацій	Журнал реєстрації інструктажів з питань охорони праці на робочому місці
Цільовий	При виконанні разових робіт, не пов'язаних з основним кодингом (переїзд офісу, ліквідація наслідків затоплення тощо)	Перед початком виконання конкретної разової технічної роботи	Наряд-допуск або фіксація у спеціалізованому журналі

Технічна та інженерна безпека офісних просторів розробників та інфраструктури веб-платформи охоплює три базові підсистеми: пожежну безпеку, електробезпеку та ергономіку мікроклімату робочого місця оператора/програміста.

Основні інженерні вимоги пожежної безпеки в ІТ-офісах та ЦОД включають:

– Забезпечення безперешкодного та постійного доступу до шляхів евакуації; двері на шляхах евакуаційних виходів не повинні мати внутрішніх засувів і мають відкриватися виключно у напрямку евакуаційного потоку;

– Оснащення приміщень розробників та особливо серверних залів автоматичними системами газового або порошкового пожежогасіння (використання водяних систем у серверних заборонено), а також високочутливими датчиками виявлення задимлення;

– Розміщення на кожному поверсі на видному місці детальних графічних схем-планів евакуації персоналу;

– Комплектацію офісних блоків первинними засобами гасіння пожеж (вуглекислотними вогнегасниками класу В та Е), які дозволяють ліквідовувати загоряння електроустановок під напругою до 1000 В.

Електробезпека при експлуатації ПЕОМ та серверних шаф вимагає безперервного моніторингу параметрів мережі. Оскільки сучасні розробницькі станції та тестові сервери платформи «FrameRate» є високопотужними споживачами електричного струму, сумарне навантаження на локальні кабельні лінії та силові шини суттєво зростає.

Технічні інженерні вимоги електробезпеки суворо регламентують:

– Обов'язкове підключення всієї техніки до трипровідної електричної мережі з окремим контуром захисного заземлення (опір заземлення не повинен перевищувати 4 Ом);

– Встановлення автоматичних пристроїв захисного відключення (ПЗО) та диференційних автоматів для захисту персоналу від витоків струму на металеві корпуси системних блоків;

– Категоричну заборону послідовного підключення мережевих подовжувачів («гірлянд») та використання пошкоджених діелектричних оболонок силових кабелів;

– Проведення планових інструментальних замірів опору ізоляції та перевірки цілісності заземлювальних провідників не рідше ніж один раз на рік.

При проєктуванні та організації фізичного робочого місця програміста вимоги ергономіки та безпеки інтегруються безпосередньо у геометрію та планування простору. Робочий стіл та крісло повинні мати механізми регулювання висоти для забезпечення оптимального кута згинання ліктьових та

колінних суглобів (близько 90-100 градусів). Відеодисплейний термінал (монітор) встановлюється на відстані 60-70 см від очей користувача, а верхня кромка екрана повинна знаходитися на рівні очей або трохи нижче для зниження статичного навантаження на шийний відділ хребта.

Дотримання цих комплексних інженерно-технічних вимог дозволяє мінімізувати вплив шкідливих виробничих факторів та забезпечити високу працездатність команди проєкту[35].

4.3 Висновок до четвертого розділу

У четвертому розділі кваліфікаційної роботи виконано комплексний технічний та нормативно-правовий аналіз питань забезпечення безпеки життєдіяльності та охорони праці під час розробки та функціонування веб-платформи електронної комерції «FrameRate».

Сформульовано чіткі, покрокові інженерно-медичні алгоритми надання невідкладної домедичної допомоги при специфічних для ІТ-галузі травмах та небезпечних станах - ураженні електричним струмом високої напруги та синкопальних станах (непритомності), що дозволяє мінімізувати ризики для здоров'я персоналу до моменту прибуття екстрених служб. Математично обґрунтовано часові обмеження безперервної експлуатації відеодисплейних терміналів за санітарно-гігієнічними вимогами[31].

Досліджено структуру та вимоги щодо проведення п'яти видів обов'язкових інструктажів з техніки безпеки на виробництві, побудовано їх класифікаційну матрицю. Визначено суворі інженерно-технічні вимоги щодо пожежної безпеки офісних блоків та серверних кімнат, а також параметри забезпечення електробезпеки при роботі з потужними обчислювальними системами онлайн-магазину моніторів.

Доведено, що інтеграція нормативних ергономічних, санітарних та інженерно-технічних вимог безпосередньо на етапі проєктування та

експлуатації програмно-апаратного комплексу дозволяє створити безпечне, інклюзивне, стійке до техногенних ризиків робоче середовище для розробників та технічного персоналу платформи «FrameRate».

ВИСНОВКИ

У кваліфікаційній роботі на основі проведених досліджень вирішено актуальне науково-практичне завдання з проєктування, програмної реалізації, розгортання та експериментального тестування високотехнологічної веб-платформи електронної комерції «FrameRate», спеціалізованої на дистрибуції моніторів преміум-сегменту.

За результатами виконання етапів розробки та дослідження сформульовано такі основні висновки та науково-практичні результати:

– Проведено комплексний системний аналіз предметної області електронної комерції (E-commerce) та індустрії роздрібної онлайн-торгівлі комп'ютерною периферією. Визначено ключові функціональні вимоги до сучасних веб-маркетплейсів, серед яких пріоритетними є висока швидкість відгуку інтерфейсу, безпека збереження даних, наявність інструментів гнучкої фільтрації та простота адміністрування контенту.

– Обґрунтовано та впроваджено сучасний технологічний стек розробки MERN (MongoDB Atlas, Express, React, Node.js). Вибір даної архітектурної комбінації дозволив реалізувати концепцію децентралізованого веб-додатка (Decoupled Architecture), у якому серверна логіка (Backend) та користувацький інтерфейс (Frontend) функціонують як автономні ізольовані системи, взаємодіючи виключно через асинхронні транзакції REST API, що суттєво підвищує загальну відмовостійкість програмного комплексу.

– Спроєктовано та розгорнуто архітектуру неструктурованої NoSQL бази даних у хмарній інфраструктурі MongoDB Atlas. Використання об'єктно-документного відображення Mongoose ODM дозволило задекларувати сувору типізацію та правила первинної валідації сутності «Монітор» (Product) на рівні сервера. Хмарна організація СКБД забезпечила глобальну доступність даних, високу швидкість обробки JSON-пакетів та масштабованість сховища без прив'язки до локальних апаратних ресурсів.

– Розроблено стійку серверну частину (Backend) на базі програмної платформи Node.js та фреймворку Express. Реалізовано архітектурні ендпоінти для обробки основних типів HTTP-запитів (GET, POST, DELETE), інтегровано модулі проміжного програмного забезпечення для парсингу вхідного трафіку та налаштовано політику CORS (Cross-Origin Resource Sharing) для безпечного міждоменного обміну даними з клієнтом.

– Створено інтерактивний користувацький інтерфейс (Frontend) за технологією SPA (Single Page Application) на базі бібліотеки React та надшвидкого збирача Vite. Завдяки впровадженню реактивних станів (React Hooks) та асинхронного HTTP-клієнта Axios, реалізовано динамічну вітрину товарів, яка функціонує без перезавантаження веб-сторінки, що мінімізує навантаження на мережу та покращує користувацький досвід (UX)[22].

– Інтегровано розширені алгоритми обробки даних на стороні клієнта (Client-Side Processing). Розроблено модулі багаторівневої інтерактивної фільтрації за технічними характеристиками (типи матриць IPS/VA/OLED, частота розгортки від 60 до 360 Hz), живого текстового пошуку за підрядком найменування/бренду, а також дворівневого сортування за вектором вартості товарів. Візуальне оформлення та стовідсотковий адаптивний дизайн (Responsive Web Design) під мобільні та широкоформатні екрани реалізовано засобами утилітарного CSS-фреймворку Tailwind CSS.

– Успішно реалізовано повний життєвий цикл операцій CRUD через адміністративний модуль та систему видалення карток, а також розроблено додаткові бізнес-компоненти: дворівневу панель оформлення замовлення (Checkout) з анкетою персональних даних доставки та інтерактивний модуль зворотного зв'язку і клієнтських відгуків з графічним рендерингом зоряного рейтингу.

– Проведено експериментальне прийомо-здавальне тестування методами Black Box розробленої системи. Результати верифікації підтвердили повну працездатність усіх функціональних блоків, відсутність логічних помилок або витоків пам'яті. Швидкість виконання внутрішніх алгоритмів

клієнтської фільтрації склала менше 5 мс, що свідчить про високу ергономічність та інженерну ефективність програмного продукту.

– У розділі охорони праці та безпеки життєдіяльності виконано детальний аналіз специфічних виробничих ризиків ІТ-галузі. Сформульовано алгоритми надання домедичної допомоги при ураженні електричним струмом та непритомності в офісних приміщеннях, проаналізовано систему інструктажів, а також визначено суворі інженерні вимоги щодо електробезпеки, пожежної безпеки та ергономіки організації робочого місця програміста. Доведено, що інтеграція санітарно-технічних норм безпосередньо на етапі проєктування середовища дозволяє звести до мінімуму вплив шкідливих виробничих факторів.

Розроблена веб-платформа електронної комерції «FrameRate» повністю відповідає вимогам затвердженого технічного завдання, має високу комерційну привабливість, інженерну завершеність і готова до практичного впровадження у реальний сектор цифрової економіки.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Онлайн-маркетплейс rozetka : офіційний сайт. Url: https://rozetka.com.ua (дата звернення: 12.05.2026).

2. Омніканальна торговельна мережа comfy : офіційний сайт. Url: https://comfy.ua (дата звернення: 14.05.2026).

3. Спеціалізований комп'ютерний центр telemart : офіційний сайт. Url: https://telemart.ua (дата звернення: 18.05.2026).

4. Плескач в. Л., завадська т. Г. Електронна комерція : підручник. Київ : знання, 2022. 582 с.

5. Азарова А. О., Малахова О. В. Сучасні тенденції розвитку ринку електронної комерції в Україні. Економіка та суспільство. 2023. № 49.Url:https://economyandsociety.in.ua/index.php/journal/article/view/2311.

6. Ux/ui дизайн для e-commerce систем: проектування інтерфейсів для високої конверсії : монографія / за ред. Д. О. Коваленка. Харків : фоліо, 2024. 314 с.

7. Гавва в. М., крамар о. В. Системний аналіз та проектування користувацького досвіду в інтернет-магазинах комп'ютерної техніки. Комп'ютерні науки та інженерія. 2024. Т. 15, № 2. С. 45–53.

8. Соммервілл і. Інженерія програмного забезпечення : підручник / пер. З англ. Київ : видавничий дім «києво-могилянська академія», 2023. 842 с.

9. Мартін р. Чиста архітектура. Структура і дизайн програмного забезпечення : практ. Посібник / пер. З англ. Львів : видавництво старого лева, 2023. 412 с.

10. Швабер к., сазерленд д. Вичерпний посібник зі скраму: правила гри : офіційне керівництво scrum.org. 2020. 22 с.

11. Небесний р. М. Рекомендаційна система формування команд виконавців з відповідними фаховими компетентностями : дис. ... Канд. Техн. Наук (або: д-ра філософії) / руслан михайлович небесний ; тернопільський

національний технічний університет імені івана пулюя. Тернопіль, 2023. Url: https://elartu.tntu.edu.ua/handle/lib/43005 (дата звернення: 17.06.2026).

12. Formation of an it project team by analogy with a flock / r. Vaskiv, n. Veretennikova, r. Nebesnyi, h. Bilovus, y. Zhovnir. 2024 ieee 19th international conference on computer science and information technologies (csit) (lviv, 16–19 oct. 2024). Lviv : ieee, 2024. P. 01–04. Url: https://doi.org/10.1109/csit65290.2024.10982684.

13. Smart people: the role of big data analytics in digital transformation / r. Nebesnyi, o. Palka, l. Dmytrotsa, h. Kozbur. Baitmp 2025: the 2nd international workshop on bioinformatics and applied information technologies for medical purpose 2025. 2026. Vol. 2. P. 163–174. Url: https://ceur-ws.org/vol-4159/paper14.pdf (дата звернення: 17.06.2026).

14. Portfolio project management / r. Nebesnyi, n. Kunanets, n. Veretennikova, r. Vaskiv, z. Haladzhun, m. Graca. Itpm. 2024. P. 141–152. Url: https://ceur-ws.org/vol-3709/paper12.pdf (дата звернення: 17.06.2026).

15. Formation of an it project team in the context of pmbok requirements / r. Nebesnyi, n. Kunanets, r. Vaskiv, n. Veretennikova. 2021 ieee 16th international conference on computer sciences and information technologies (csit). 2021. Vol. 2. P. 431–436. Url: https://doi.org/10.1109/csit52704.2021.9615291 (дата звернення: 17.06.2026).

16. Фленаган д. Javascript: детальний посібник : інженерний курс. 7-ме вид. Львів : бакалай, 2022. 760 с.

17. Чініатхамбі к. Знайомство з react. Створення сучасних single page applications : навчальний посібник. Дніпро : інновація, 2023. 368 с.

18. Banks a., porcello e. Learning react: modern patterns for developing real-world applications. 2nd ed. Sebastopol : o'reilly media, 2020. 320 p.

19. Спад п. Розробка додатків з використанням redux toolkit та сучасного react : практ. Курс. Київ : комп’ютерпрес, 2024. 284 с.

20. Гевенд с. Розробка веб-додатків на node.js та express.js : професійний посібник / пер. З англ. Київ : дія, 2023. 496 с.

21. Тейшейра п. Професійна розробка на node.js. Асинхронний event loop та побудова rest api. Харків : ранок, 2023. 344 с.

22. Бенкер к. Mongodb в дії. Робота з нереляційними базами даних nosql : інженерне керівництво. Одеса : купрієнко, 2022. 512 с.

23. Чодоров к. Повне керівництво з хмарної субд mongodb atlas : навчальний посібник. Київ : мультимедіа, 2023. 294 с.

24. Холмс к. Розробка веб-додатків за допомогою стеку mern: mongodb, express, react, node.js. Львів : світ, 2024. 612 с.

25. Швайгер с. Безпека веб-додатків: авторизація jwt, захист паролів за допомогою bcrypt та налаштування cors. Журнал кібербезпеки та програмної інженерії. 2023. № 4. С. 112–121.

26. Родрігес а. Проектування restful веб-сервісів за допомогою http-клієнта axios. Комп'ютерні технології та програмування. 2024. Т. 8, № 1. С. 78–85.

27. Tailwind css: сучасний підхід до компонентної стилізації single page applications : методичні вказівки / уклад. О. В. Павленко. Тернопіль : тнту, 2024. 48 с.

28. Попов о. В. Оптимізація та конвеєризація збірки веб-додатків за допомогою vite. Сучасна інженерія програмного забезпечення. 2024. № 14. С. 34–42.

29. Оцінка продуктивності та seo-оптимізація веб-ресурсів засобами google lighthouse : практичний посібник / за ред. І. О. Боднарчука. Тернопіль : палітра, 2025. 124 с.

30. Лайза к. Тестування програмного забезпечення за методом black box: практичні сценарії. Київ : техніка, 2023. 216 с.

31. Про охорону праці : закон україни від 14.10.1992 № 2694-xii : станом на 2025 р. Url: https://zakon.rada.gov.ua/laws/show/2694-12 (дата звернення: 02.06.2026).

32. Кодекс законів про працю україни : закон україни від 10.12.1971 № 322-viii : станом на 2026 р. Url: https://zakon.rada.gov.ua/laws/show/322-08 (дата звернення: 02.06.2026).

33. Про затвердження порядків надання домедичної допомоги особам при невідкладних станах : наказ міністерства охорони здоров'я україни від 09.03.2022 № 441. Офіційний вісник україни. 2022. № 25. Стаття 135. Url: https://zakon.rada.gov.ua/laws/show/z0395-22 (дата звернення: 03.06.2026).

34. Купчак в. Р., олійник л. В. Домедична допомога при виробничому та побутовому травматизмі : практичний посібник. Львів : новий світ-2000, 2023. 142 с.

35. Нпаоп 0.00-4.12-05. Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці : чинне від 2023 р. Київ : держпраці, 2023. 45 с.

36. Нпаоп 0.00-7.15-18. Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з візуальними дисплейними терміналами. Київ : мінсоцполітики україни, 2018. 24 с.

37. Дбн в.1.1-7:2016. Пожежна безпека об'єктів будівництва. Основні положення : чинне від 2022 р. Київ : мінрегіон україни, 2022. 52 с.

38. Нпаоп 40.1-1.21-98. Правила безпечної експлуатації електроустановок споживачів : чинна редакція від 2022 р. Київ : держнаглядохоронпраці, 2022. 68 с.

39. Дсн 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень : актуалізовано станом на 2024 р. Київ : моз україни, 2024. 18 с.

40. Ткачук к. Н., халімовський м. О. Основи охорони праці в іт-галузі та сферах комерційних послуг : навчальний посібник. Київ : кпі ім. Ігоря сікорського, 2023. 185 с.

ДОДАТКИ

Додаток А

Приклади коду серверної частини сайту

Лістинг А1 - вміст файлу index.js

```
const express = require('express');
const cors = require('cors');
const Monitor = require('./models/product.model');

const app = express();
app.use(cors());
app.use(express.json());

app.get('/api/products', async (req, res) => {
  try {
    let query = {};
    if (req.query.panelType && req.query.panelType !== 'all') {
      query.panelType = req.query.panelType;
    }
    if (req.query.refreshRate && req.query.refreshRate !== 'all') {
      query.refreshRate = parseInt(req.query.refreshRate);
    }
    const monitors = await Monitor.find(query);
    res.json(monitors);
  } catch (error) {
    res.status(500).json({ message: 'Помилка сервера' });
  }
});

app.post('/api/products', async (req, res) => {
  try {
    const newMonitor = new Monitor(req.body);
    await newMonitor.save();
    res.status(201).json(newMonitor);
  } catch (error) {
    console.error("❌ КРИТИЧНА ПОМИЛКА БАЗИ ДАНИХ:", error.message);

    res.status(400).json({ message: 'Некоректні дані' });
  }
});

app.delete('/api/products/:id', async (req, res) => {
  try {
    await Monitor.findByIdAndDelete(req.params.id);
    res.json({ message: 'Товар успішно видалено з бази даних' });
  } catch (error) {
    res.status(500).json({ message: 'Помилка при видаленні товару' });
  }
});

module.exports = { app };
```

Лістинг А2 – вміст файлу server.js

```
console.log("=== ТЕСТ: Node.js успішно прочитав цей файл! ===");

const { app } = require('./index');
const { connectToDatabase } = require('./config/db');

const PORT = 5000;

const startServer = async () => {
  await connectToDatabase();
  app.listen(PORT, () => {
    console.log(`Бекенд сервер успішно запущено на порту ${PORT}`);
  });
};

startServer();
```

Лістинг А3 – вміст файлу config/db.js

```
// backend/src/config/db.js
const mongoose = require('mongoose');

const dbURI =
    'mongodb+srv://frame:frame123@cluster0.ytbo8nq.mongodb.net/fra
    merate_db?appName=Cluster0';

const connectToDatabase = async () => {
  try {
    await mongoose.connect(dbURI);
    console.log('Успішне підключення до ХМАРНОЇ бази даних MongoDB
    Atlas!');
  } catch (err) {
    console.error('Помилка підключення до хмарної БД:', err);
    process.exit(1);
  }
};

module.exports = { connectToDatabase };
```

Лістинг А4 – вміст файлу models/productmodels.js

```
const mongoose = require('mongoose');

const monitorSchema = new mongoose.Schema({
  title: { type: String, required: true },
  brand: { type: String, required: true },
  price: { type: Number, required: true },
   panelType: { type: String, enum: ['IPS', 'VA', 'OLED'],
    required: true },
  refreshRate: { type: Number, required: true },
```

```
  imageUrl: { type: String }
});

module.exports = mongoose.model('products', monitorSchema);
```

Лістинг А5 - вміст файлу

```
{
  "name": "backend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "cors": "^2.8.6",
    "dotenv": "^17.4.2",
    "express": "^5.2.1",
    "mongoose": "^9.7.0"
  }
}
```

Додаток Б

Приклади коду клієнтської частини сайту

Лістинг Б1 – вміст файлу App.js

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';

export default function App() {
  const [monitors, setMonitors] = useState([]);
  const [panelType, setPanelType] = useState('all');
  const [refreshRate, setRefreshRate] = useState('all');
  const [cart, setCart] = useState([]);
  const [isCartOpen, setIsCartOpen] = useState(false);

  // Фільтрація, сортування та ПОШУК
  const [selectedBrand, setSelectedBrand] = useState('all');
  const [sortBy, setSortBy] = useState('none');
   const [searchQuery, setSearchQuery] = useState(''); // Новий
    стан для рядка пошуку

  // Оформлення замовлення
  const [isCheckoutMode, setIsCheckoutMode] = useState(false);
  const [customerName, setCustomerName] = useState('');
  const [customerPhone, setCustomerPhone] = useState('');
    const [deliveryMethod, setDeliveryMethod] = useState('Нова
    Пошта');
  const [deliveryAddress, setDeliveryAddress] = useState('');
    const [paymentMethod, setPaymentMethod] = useState('При
    отриманні');

  // Адмін-панель
  const [title, setTitle] = useState('');
  const [brand, setBrand] = useState('');
  const [price, setPrice] = useState('');
  const [imageUrl, setImageUrl] = useState('');
  const [adminPanel, setAdminPanel] = useState('IPS');
  const [adminRefresh, setAdminRefresh] = useState(144);

  // Система відгуків
    const [activeReviewMonitorId, setActiveReviewMonitorId] =
    useState(null);
  const [reviewName, setReviewName] = useState('');
  const [reviewRating, setReviewRating] = useState(5);
  const [reviewText, setReviewText] = useState('');

  const [reviews, setReviews] = useState({
    default: [
        { name: "Владислав", rating: 5, text: "Колірна передача
    просто неймовірна, для роботи з графікою те що треба!" },
```

```
      { name: "Артем", rating: 4, text: "Герцовка відчувається з
  перших секунд в іграх. Якість збірки на висоті." }
    ]
});

const fetchMonitors = async () => {
  try {
                          const       response       =       await
  axios.get(`http://localhost:5000/api/products`, {
      params: { panelType, refreshRate }
    });
    setMonitors(response.data);
  } catch (error) {
    console.error("Помилка при завантаженні моніторів:", error);
  }
};

useEffect(() => {
  fetchMonitors();
}, [panelType, refreshRate]);

const handleDeleteMonitor = async (id) => {
    if  (!window.confirm('Ви впевнені, що хочете видалити цей
  монітор з бази даних?')) return;
  try {
                                                              await
  axios.delete(`http://localhost:5000/api/products/${id}`);
    alert('Товар успішно видалено!');
    fetchMonitors();
  } catch (error) {
    alert('Помилка при видаленні товару');
  }
};

const addToCart = (monitor) => {
  setCart([...cart, monitor]);
};

const handleConfirmOrder = (e) => {
  e.preventDefault();
  if (!customerName || !customerPhone || !deliveryAddress) {
     return alert('Будь ласка, заповніть усі поля для оформлення
  доставки!');
  }
   alert(`🎉 Замовлення №${Math.floor(Math.random() * 90000) +
  10000} успішно сформовано!`);
      setCart([]);   setCustomerName('');   setCustomerPhone('');
  setDeliveryAddress('');
  setIsCheckoutMode(false); setIsCartOpen(false);
};

const handleAddMonitor = async (e) => {
  e.preventDefault();
```

```
    if (!title || !price || !brand) return alert('Будь лас
  заповніть обов'язкові поля!');
          const finalImageUrl = imageUrl.trim() ||
  'https://images.unsplash.com/photo-1527443224154-
  c4a3942d3acf?w=500';

  try {
    await axios.post('http://localhost:5000/api/products', {
      title,
      brand,
      price: Number(price),
      panelType: adminPanel,
      refreshRate: Number(adminRefresh),
      imageUrl: finalImageUrl
    });

    alert('Нову модель монітора успішно додано!');
    setTitle(''); brand(''); price(''); setImageUrl('');
    fetchMonitors();
  } catch (error) {
    alert('Помилка при збереженні товару');
  }
};

const handleAddReview = (e) => {
  e.preventDefault();
    if (!reviewName || !reviewText) return alert('Будь ласка,
  заповніть ім'я та текст відгуку!');
  const targetId = activeReviewMonitorId || 'default';
      const currentMonitorReviews = reviews[targetId] ||
  [...(reviews['default'] || [])];

  setReviews({
    ...reviews,
            [targetId]: [{ name: reviewName, rating:
  Number(reviewRating), text: reviewText },
  ...currentMonitorReviews]
  });

  alert('Дякуємо! Ваш відгук успішно опубліковано.');
  setReviewName(''); setReviewText(''); setReviewRating(5);
};

// ==========================================
// ОБРОБКА ДАНИХ + ФІЛЬТР ЖИВОГО ПОШУКУ
// ==========================================
const filteredAndSortedMonitors = monitors
  .filter(monitor => {
    if (selectedBrand === 'all') return true;
                return monitor.brand.toLowerCase() ===
  selectedBrand.toLowerCase();
  })
```

```
  // ДРУГИЙ КРОК: Фільтрація за текстом з пошукового рядка (шукає по назві та бренду)
  .filter(monitor => {
              const searchFields = `${monitor.title} ${monitor.brand}`.toLowerCase();
    return searchFields.includes(searchQuery.toLowerCase());
  })
  .sort((a, b) => {
    if (sortBy === 'lowToHigh') return a.price - b.price;
    if (sortBy === 'highToLow') return b.price - a.price;
    return 0;
  });

 const totalCartPrice = cart.reduce((sum, item) => sum + item.price, 0);

return (
   <div className="min-h-screen bg-gray-50 text-gray-800 font-sans relative">

    {/* Шапка сайту */}
      <header className="bg-slate-950 text-white p-4 shadow-xl sticky top-0 z-40 border-b border-slate-800">
        <div className="container mx-auto flex justify-between items-center px-4">
        <div>
              <h1 className="text-3xl font-black tracking-wider text-blue-500">FrameRate</h1>
               <p className="text-xs font-medium text-gray-400 uppercase tracking-widest">Premium Monitor Store</p>
        </div>

        <button
                  onClick={() => { setIsCartOpen(!isCartOpen); setIsCheckoutMode(false); }}
            className="bg-blue-600 hover:bg-blue-700 text-white font-bold px-5 py-2.5 rounded-xl transition flex items-center gap-2 relative shadow-lg shadow-blue-900/20"
        >
          🛒 Кошик
           <span className="bg-white text-blue-600 text-xs font-black px-2 py-0.5 rounded-full">
             {cart.length}
           </span>
        </button>
      </div>
    </header>

    {/* Головний простір */}
    <div className="container mx-auto p-4 lg:p-8 grid grid-cols-1 lg:grid-cols-4 gap-8">

      {/* Бічна панель: Пошук, Фільтри та Сортування */}
```

```
        <aside className="bg-white p-6 rounded-2xl shadow-sm border border-gray-100 h-fit space-y-5">
          <h2 className="text-lg font-extrabold text-gray-900 border-b pb-3 flex items-center gap-2">
        ⚙️ Панель навігації
      </h2>

      {/* НОВИЙ БЛОК: Текстовий пошук */}
         <div className="bg-blue-50/40 p-3 rounded-xl border border-blue-100/60">
          <label className="block text-xs font-black uppercase tracking-wider text-blue-600 mb-1.5">Пошук по каталогу</label>
        <input
          type="text"
          value={searchQuery}
          onChange={(e) => setSearchQuery(e.target.value)}
          placeholder="Введіть модель або бренд..."
           className="w-full p-2.5 bg-white border border-gray-200 rounded-lg text-xs font-medium focus:outline-none focus:ring-2 focus:ring-blue-500 transition shadow-xs text-slate-700 placeholder:text-gray-400"
        />
      </div>

      <div>
          <label className="block text-xs font-bold uppercase tracking-wider text-gray-400 mb-2">Сортування за ціною</label>
                <select value={sortBy} onChange={(e) => setSortBy(e.target.value)} className="w-full p-3 bg-gray-50 border border-gray-200 rounded-xl font-medium focus:outline-none focus:ring-2 focus:ring-blue-500 transition text-sm text-slate-700">
          <option value="none">За замовчуванням</option>
               <option value="lowToHigh">💰 Від дешевшої до дорожчої</option>
               <option value="highToLow">💎 Від дорожчої до дешевшої</option>
        </select>
      </div>

      <div>
          <label className="block text-xs font-bold uppercase tracking-wider text-gray-400 mb-2">Виробник монітора</label>
             <select value={selectedBrand} onChange={(e) => setSelectedBrand(e.target.value)} className="w-full p-3 bg-gray-50 border border-gray-200 rounded-xl font-medium focus:outline-none focus:ring-2 focus:ring-blue-500 transition text-sm text-slate-700">
          <option value="all">Усі виробники</option>
          <option value="MSI">MSI</option>
          <option value="Acer">Acer</option>
          <option value="Gigabyte">Gigabyte</option>
```

```
          <option value="Asus">Asus</option>
          <option value="Samsung">Samsung</option>
        </select>
      </div>

      <div>
          <label className="block text-xs font-bold uppercase tracking-wider text-gray-400 mb-2">Тип матриці</label>
                <select value={panelType} onChange={(e) => setPanelType(e.target.value)} className="w-full p-3 bg-gray-50 border border-gray-200 rounded-xl font-medium focus:outline-none focus:ring-2 focus:ring-blue-500 transition text-sm">
          <option value="all">Усі типи панелей</option>
          <option value="IPS">IPS (Точна передача)</option>
          <option value="VA">VA (Глибокий контраст)</option>
                                <option value="OLED">OLED (Кінематографічна)</option>
        </select>
      </div>

      <div>
          <label className="block text-xs font-bold uppercase tracking-wider text-gray-400 mb-2">Частота розгортки</label>
              <select value={refreshRate} onChange={(e) => setRefreshRate(e.target.value)} className="w-full p-3 bg-gray-50 border border-gray-200 rounded-xl font-medium focus:outline-none focus:ring-2 focus:ring-blue-500 transition text-sm">
          <option value="all">Будь-яка частота</option>
          <option value="60">60 Hz</option>
          <option value="144">144 Hz</option>
          <option value="240">240 Hz</option>
          <option value="360">360 Hz</option>
        </select>
      </div>
    </aside>

    {/* Вітрина товарів */}
    <main className="lg:col-span-3">
      <div className="flex justify-between items-center mb-6">
            <h2 className="text-2xl font-black text-gray-900 tracking-tight">Доступні моделі моніторів</h2>
           <span className="text-sm font-bold bg-gray-200/60 text-gray-600 px-3 py-1 rounded-full">Знайдено: {filteredAndSortedMonitors.length}</span>
      </div>

      {filteredAndSortedMonitors.length === 0 ? (
         <div className="bg-white p-12 rounded-3xl text-center border border-gray-100 shadow-sm">
           <p className="text-gray-400 text-lg font-medium">За вашим текстовим запитом або фільтрами нічого не знайдено.</p>
        </div>
```

```
      ) : (
              <div  className="grid  grid-cols-1  md:grid-cols-2
xl:grid-cols-3 gap-6">
          {filteredAndSortedMonitors.map(monitor => {
               const monitorReviews = reviews[monitor._id] ||
reviews['default'];
            return (
                  <div  key={monitor._id}  className="bg-white
rounded-2xl p-4 shadow-sm border border-gray-100 flex flex-col
justify-between      hover:shadow-xl      hover:border-blue-100
transition duration-300 group relative">

               <button
                                                  onClick={()    =>
handleDeleteMonitor(monitor._id)}
                   className="absolute top-3 right-3 bg-red-500
hover:bg-red-600 text-white w-7 h-7 rounded-lg shadow-md flex
items-center   justify-center   transition   opacity-0   group-
hover:opacity-100 z-10 text-xs font-bold"
                >
                  🗑
                </button>

                <div className="overflow-hidden rounded-xl mb-
4 bg-gray-50 flex items-center justify-center h-44">
                                     <img   src={monitor.imageUrl}
alt={monitor.title} className="w-full h-full object-contain p-
2 group-hover:scale-105 transition duration-300" />
                </div>

                <div className="flex-grow">
                    <div className="flex justify-between items-
center mb-2">
                     <div className="flex gap-1.5">
                         <span className="text-[10px] font-black
uppercase tracking-wider bg-blue-50 text-blue-600 px-2 py-0.5
rounded-md">{monitor.panelType}</span>
                         <span className="text-[10px] font-black
uppercase tracking-wider bg-amber-50 text-amber-700 px-2 py-
0.5 rounded-md">{monitor.refreshRate} Hz</span>
                     </div>
                                     <button   onClick={()   =>
setActiveReviewMonitorId(monitor._id)}       className="text-xs
font-bold text-blue-600 hover:underline flex items-center gap-
1">
                       💬 Відгуки ({monitorReviews.length})
                     </button>
                   </div>

                    <h3 className="text-lg font-bold text-gray-
900           leading-snug           group-hover:text-blue-600
transition">{monitor.title}</h3>
```

```
                    <p className="text-xs font-bold text-gray-
400       mt-1       uppercase       tracking-wider">Виробник:
{monitor.brand}</p>
                </div>

                    <div className="flex justify-between items-
center mt-5 pt-3 border-t border-gray-50">
                    <span className="text-2xl font-black text-
gray-900">{monitor.price.toLocaleString()}                <span
className="text-sm font-bold text-gray-500">грн</span></span>
                    <button onClick={() => addToCart(monitor)}
className="bg-slate-900 hover:bg-blue-600 text-white text-xs
font-black px-4 py-2.5 rounded-xl transition duration-200
shadow-md">
                    + Купити
                  </button>
                </div>
              </div>
            );
          })}
        </div>
      )}

      {/* Панель адміністратора CRUD */}
        <section className="mt-14 bg-slate-900 text-white p-6
lg:p-8 rounded-3xl shadow-xl border border-slate-800">
        <div className="border-b border-slate-800 pb-4 mb-6">
              <h2 className="text-xl font-black tracking-wide
text-amber-400 uppercase">💻 Модуль Адміністрування (Код
CRUD)</h2>
            <p className="text-xs font-medium text-gray-400 mt-
1">Додавання нових одиниць моніторів до бази даних MongoDB
Atlas в реальному часі</p>
        </div>

            <form onSubmit={handleAddMonitor} className="grid
grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-5">
          <div>
              <label className="block text-xs font-bold text-
gray-400 mb-1.5 uppercase tracking-wider">Модель монітора
*</label>
              <input type="text" value={title} onChange={(e) =>
setTitle(e.target.value)}  placeholder="Напр.  UltraGear  27"
className="w-full p-2.5 rounded-xl bg-slate-800 border border-
slate-700 text-white text-sm focus:outline-none focus:ring-2
focus:ring-amber-400" />
          </div>
          <div>
              <label className="block text-xs font-bold text-
gray-400 mb-1.5 uppercase tracking-wider">Бренд *</label>
                    <select value={brand} onChange={(e) =>
setBrand(e.target.value)} className="w-full p-2.5 rounded-xl
bg-slate-800  border  border-slate-700  text-white  text-sm
```

```
focus:outline-none focus:ring-2 focus:ring-amber-400 font-bold">
              <option value="">Оберіть бренд...</option>
              <option value="MSI">MSI</option>
              <option value="Acer">Acer</option>
              <option value="Gigabyte">Gigabyte</option>
              <option value="Asus">Asus</option>
              <option value="Samsung">Samsung</option>
            </select>
          </div>
          <div>
              <label className="block text-xs font-bold text-gray-400 mb-1.5 uppercase tracking-wider">Ціна (грн) *</label>
              <input type="number" value={price} onChange={(e) => setPrice(e.target.value)} placeholder="14500" className="w-full p-2.5 rounded-xl bg-slate-800 border border-slate-700 text-white text-sm focus:outline-none focus:ring-2 focus:ring-amber-400" />
          </div>
          <div className="lg:col-span-2">
              <label className="block text-xs font-bold text-gray-400 mb-1.5 uppercase tracking-wider">URL-посилання на зображення</label>
              <input type="text" value={imageUrl} onChange={(e) => setImageUrl(e.target.value)} placeholder="Вставте посилання https://..." className="w-full p-2.5 rounded-xl bg-slate-800 border border-slate-700 text-white text-sm focus:outline-none focus:ring-2 focus:ring-amber-400" />
          </div>
          <div>
              <label className="block text-xs font-bold text-gray-400 mb-1.5 uppercase tracking-wider">Характеристики</label>
            <div className="flex gap-2">
                <select value={adminPanel} onChange={(e) => setAdminPanel(e.target.value)} className="w-1/2 p-2.5 rounded-xl bg-slate-800 border border-slate-700 text-white text-xs font-bold focus:outline-none">
                <option value="IPS">IPS</option>
                <option value="VA">VA</option>
                <option value="OLED">OLED</option>
              </select>
                <select value={adminRefresh} onChange={(e) => setAdminRefresh(e.target.value)} className="w-1/2 p-2.5 rounded-xl bg-slate-800 border border-slate-700 text-white text-xs font-bold focus:outline-none">
                <option value="60">60Hz</option>
                <option value="144">144Hz</option>
                <option value="240">240Hz</option>
                <option value="360">360Hz</option>
              </select>
            </div>
          </div>
```

```
            <div className="lg:col-span-3 flex justify-end mt-
2">
                <button type="submit" className="w-full lg:w-48
bg-amber-500 hover:bg-amber-600 text-slate-950 font-black p-3
text-xs uppercase tracking-wider rounded-xl transition
duration-200 shadow-lg">
                  💾 Зберегти товар
                </button>
              </div>
            </form>
          </section>
        </main>
      </div>

      {/* Модальне вікно відгуків */}
      {activeReviewMonitorId && (
        <div className="fixed inset-0 bg-black/60 backdrop-blur-xs
z-50 flex items-center justify-center p-4">
          <div className="bg-white rounded-3xl max-w-2xl w-full
max-h-[85vh] flex flex-col p-6 shadow-2xl relative">
            <div className="flex justify-between items-center
border-b pb-3 mb-4">
              <div>
                <h3 className="text-xl font-black text-gray-
900">💬 Відгуки та оцінки покупців</h3>
                <p className="text-xs font-bold text-gray-400
uppercase mt-0.5">Модель: {monitors.find(m => m._id ===
activeReviewMonitorId)?.title || 'Монітор'}</p>
              </div>
              <button onClick={() => {
setActiveReviewMonitorId(null); setReviewName('');
setReviewText(''); }} className="text-gray-400 hover:text-
gray-600 font-black text-lg">✕</button>
            </div>
            <div className="flex-grow overflow-y-auto space-y-3
mb-4 pr-1">
              {(reviews[activeReviewMonitorId] ||
reviews['default']).map((rev, idx) => (
                <div key={idx} className="p-4 bg-gray-50 rounded-
2xl border border-gray-100">
                  <div className="flex justify-between items-
center mb-1">
                    <span className="font-black text-sm text-gray-
900">{rev.name}</span>
                    <span className="text-amber-500 font-bold
text-sm">{"⭐".repeat(rev.rating)}</span>
                  </div>
                  <p className="text-sm text-gray-600 leading-
relaxed font-medium">{rev.text}</p>
                </div>
              ))}
            </div>
```

```
          <form onSubmit={handleAddReview} className="border-t
pt-4 space-y-3 bg-white">
             <h4 className="text-sm font-black text-slate-900
uppercase tracking-wider">✍️ Залишити свій відгук:</h4>
          <div className="grid grid-cols-3 gap-3">
            <div className="col-span-2">
                <input type="text" required value={reviewName}
onChange={(e)          =>          setReviewName(e.target.value)}
placeholder="Ваше ім'я" className="w-full p-2.5 rounded-xl bg-
gray-50    border    border-gray-200    text-xs    font-medium
focus:outline-none focus:ring-2 focus:ring-blue-500" />
            </div>
            <div>
                 <select value={reviewRating} onChange={(e) =>
setReviewRating(e.target.value)}    className="w-full    p-2.5
rounded-xl bg-gray-50 border border-gray-200 text-xs font-bold
text-amber-600 focus:outline-none">
                <option value="5">⭐⭐⭐⭐⭐ (5)</option>
                <option value="4">⭐⭐⭐⭐ (4)</option>
                <option value="3">⭐⭐⭐ (3)</option>
              </select>
            </div>
          </div>
          <div>
               <textarea required rows="2" value={reviewText}
onChange={(e)          =>          setReviewText(e.target.value)}
placeholder="Напишіть вашу думку про цю модель монітора..."
className="w-full p-2.5 rounded-xl bg-gray-50 border border-
gray-200 text-xs font-medium focus:outline-none focus:ring-2
focus:ring-blue-500 resize-none" />
          </div>
          <div className="flex justify-end">
                 <button type="submit" className="bg-blue-600
hover:bg-blue-700 text-white font-black px-6 py-2.5 rounded-xl
text-xs      uppercase      tracking-wider      transition">🚀
Опублікувати</button>
          </div>
        </form>
      </div>
    </div>
  )}

  {/* Кошик та Оформлення */}
  {isCartOpen &&  (
     <div className="fixed inset-0 bg-black/60 backdrop-blur-xs
z-50 flex justify-end transition duration-300">
        <div className={`bg-white h-full p-6 shadow-2xl flex
flex-col  justify-between  transition-all  duration-300  w-full
${isCheckoutMode ? 'max-w-xl' : 'max-w-md'}`}>
          <div>
              <div className="flex justify-between items-center
border-b pb-4 mb-4">
```

```
                <h2 className="text-xl font-black text-gray-
900">{isCheckoutMode ? '📋 Оформлення замовлення' : '🛒 Ваш
кошик покупок'}</h2>
                <button onClick={() => { setIsCartOpen(false);
setIsCheckoutMode(false);      }}      className="text-gray-400
hover:text-gray-600 font-bold text-xl">✕</button>
            </div>
            {cart.length === 0 ? (
                <p className="text-gray-400 text-center my-12
font-medium">Кошик порожній.</p>
            ) : !isCheckoutMode ? (
                <div className="space-y-3 overflow-y-auto max-h-
[65vh] pr-1">
                {cart.map((item, idx) => (
                    <div key={idx} className="flex justify-between
items-center p-3 bg-gray-50 rounded-xl border border-gray-
100">
                        <div>
                            <h4 className="font-bold text-sm text-
gray-900">{item.title}</h4>
                            <span className="text-[10px] font-bold
text-gray-400 uppercase">{item.panelType} •
{item.refreshRate}Hz</span>
                        </div>
                        <span className="font-black text-sm text-
gray-900">{item.price.toLocaleString()} грн</span>
                    </div>
                ))}
                </div>
            ) : (
                <form onSubmit={handleConfirmOrder}
className="space-y-4 overflow-y-auto max-h-[70vh] pr-1">
                <div className="bg-blue-50/50 border border-
blue-100 rounded-xl p-3 text-sm flex justify-between items-
center">
                    <span className="font-medium text-slate-
600">Обрано товарів: <strong className="text-blue-
600">{cart.length} шт.</strong></span>
                    <span className="font-bold text-slate-
900">Сума: {totalCartPrice.toLocaleString()} грн</span>
                </div>
                <div>
                    <label className="block text-xs font-bold
text-gray-400 mb-1 uppercase tracking-wider">Прізвище та Ім'я
*</label>
                    <input type="text" required
value={customerName} onChange={(e) =>
setCustomerName(e.target.value)} placeholder="Напр. Іванов
Сергій" className="w-full p-2.5 rounded-xl bg-gray-50 border
border-gray-200 text-sm focus:outline-none focus:ring-2
focus:ring-blue-500 font-medium" />
                </div>
```

```
            <div>
                  <label className="block text-xs font-bold
text-gray-400 mb-1 uppercase tracking-wider">Контактний
телефон *</label>
                                 <input type="tel" required
value={customerPhone}              onChange={(e)
=>
setCustomerPhone(e.target.value)} placeholder="+38 (097) 123-
4567" className="w-full p-2.5 rounded-xl bg-gray-50 border
border-gray-200 text-sm focus:outline-none focus:ring-2
focus:ring-blue-500 font-medium" />
            </div>
            <div className="grid grid-cols-2 gap-3">
              <div>
                   <label className="block text-xs font-bold
text-gray-400 mb-1 uppercase tracking-wider">Доставка</label>
                  <select value={deliveryMethod} onChange={(e)
=> setDeliveryMethod(e.target.value)} className="w-full p-2.5
rounded-xl bg-gray-50 border border-gray-200 text-sm
focus:outline-none focus:ring-2 focus:ring-blue-500 font-bold
text-slate-700">
                                <option value="Нова Пошта">Нова
Пошта</option>
                                                    <option
value="Самовивіз">Самовивіз</option>
                  </select>
                </div>
                <div>
                   <label className="block text-xs font-bold
text-gray-400 mb-1 uppercase tracking-wider">Оплата</label>
                  <select value={paymentMethod} onChange={(e)
=> setPaymentMethod(e.target.value)} className="w-full p-2.5
rounded-xl bg-gray-50 border border-gray-200 text-sm
focus:outline-none focus:ring-2 focus:ring-blue-500 font-bold
text-slate-700">
                               <option value="При отриманні">При
отриманні</option>
                    <option value="Карткою на сайті">Карткою
на сайті</option>
                  </select>
                </div>
            </div>
            <div>
                  <label className="block text-xs font-bold
text-gray-400 mb-1 uppercase tracking-wider">Місто та №
відділення доставки *</label>
                                <input type="text" required
value={deliveryAddress}            onChange={(e)
=>
setDeliveryAddress(e.target.value)}   placeholder="Напр.   м.
Тернопіль, Відділення №4" className="w-full p-2.5 rounded-xl
bg-gray-50 border border-gray-200 text-sm focus:outline-none
focus:ring-2 focus:ring-blue-500 font-medium" />
            </div>
```

```
                    <button type="submit" id="hidden-submit-btn"
    className="hidden" />
                </form>
              )}
            </div>

            {cart.length > 0 && (
              <div className="border-t pt-4">
                 <div className="flex justify-between items-center
    mb-4">
                   <span className="text-gray-500 font-bold">Разом
    до сплати:</span>
                   <span className="text-2xl font-black text-blue-
    600">{totalCartPrice.toLocaleString()} грн</span>
                </div>
                {!isCheckoutMode ? (
                   <button onClick={() => setIsCheckoutMode(true)}
    className="w-full bg-blue-600 hover:bg-blue-700 text-white
    font-black py-3 rounded-xl uppercase tracking-wider text-sm
    transition">
                     💳 Перейти до оформлення
                   </button>
                ) : (
                   <div className="flex gap-3">
                              <button type="button" onClick={() =>
    setIsCheckoutMode(false)}     className="w-1/3     bg-gray-100
    hover:bg-gray-200 text-slate-700 font-bold py-3 rounded-xl
    text-xs uppercase tracking-wider transition">
                     ⬅️ Назад
                   </button>
                                        <button   onClick={()   =>
    document.getElementById('hidden-submit-btn').click()}
    className="w-2/3 bg-emerald-500 hover:bg-emerald-600 text-
    white font-black py-3 rounded-xl uppercase tracking-wider
    text-xs transition">
                     ✓ Підтвердити
                   </button>
                   </div>
                )}
              </div>
            )}
          </div>
        </div>
      )}

    </div>
  );
}
```

Лістинг Б2 – вміст файлу index.js

```
@tailwind base;
```

```
@tailwind components;
@tailwind utilities;
```

Лістинг Б3 – вміст файлу index.jsx

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App.jsx';
import './index.css';

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

Лістинг Б4 – вміст файлу index.html

```
<!doctype html>
<html lang="uk">
  <head>
    <meta charset="UTF-8" />
       <meta   name="viewport"   content="width=device-width,   initial-
    scale=1.0" />
    <title>FrameRate — Інтернет-магазин моніторів</title>
  </head>
  <body class="bg-gray-50">
    <div id="root"></div>

    <script type="module" src="/src/index.jsx"></script>
  </body>
</html>
```

Лістинг Б5 – вміст файлу tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

Лістинг Б6 – вміст файлу vite.config.js

```
import { defineConfig } from 'vite'
```

```
import react from '@vitejs/plugin-react'

export default defineConfig({
  plugins: [react()],
})
```