

БАКАЛАВРСЬКА РОБОТА

БР. ІІ - 07.00.00.000 ІЗ

ІІ-22-4

Михайлюк Андрій

2026

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Михайлюк Андрій Васильович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка масштабованого Full-Stack застосунку для продажу комп'ютерних ігор

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121– Інженерія програмного забезпечення

(шифр і назва спеціальності)

Робота містить результати власних досліджень, використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело:

Здобувач освітнього ступеня Михайлюк А.В.
(підпис, ініціали та прізвище здобувача)

Науковий керівник Дмитрик Тарас Богданович, асистент
(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту
Завідувач кафедри

доц. Бандура В.В.
(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2026

Інститут, факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедрою ІПЗ

доц. В.В. Бандура

“ ” 2026 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Михайлюку Андрію Васильовичу

(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) “ Розробка масштабованого Full-Stack застосунку для продажу комп'ютерних ігор ”

керівник проекту (роботи) Дмитрик Тарас Богданович, асистент

затверджені наказом закладу вищої освіти від “ 21 ” квітня 2026 р. № 179/7

2. Строк подання студентом проекту (роботи) 10 червня 2026 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

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

1. Аналіз предметної області та існуючих рішень в сфері електронної комерції

2. Формування функціональних та нефункціональних вимог

3. Проектування архітектури веб-застосунку

4. Реалізація користувацької та адміністративної частин застосунку

5. Реалізація аналітичних модулів

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. Діаграма варіантів використання системи інтернет-магазину сумок (рис. 2.1, с. 28)

2. Діаграма взаємодії адміністратора з адміністративною панеллю (рис. 2.2, с. 29)

3. Діаграма діяльності процесу оформлення замовлення (рис. 2.3, с. 34)

4. Архітектура веб-застосунку інтернет-магазину (рис. 3.1, с. 36)

5. Структура взаємодії компонентів за шаблоном MVC (рис. 3.2, с. 37)

1. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

2. Дата видачі завдання 2026 року

Керівник

(підпис)

(розшифровка підпису)

Завдання прийняв до виконання

(підпис)

(розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Визначення та обґрунтування теми бакалаврської роботи	15.01.2026	Виконано
2	Проектування архітектури та системні вимоги віртуального порталу типу "GameShop"	20.01.2026	Виконано
3	Проектування архітектури веб-застосунку, структури бази даних та інтерфейсу	28.02.2026	Виконано
4	Реалізація функціональних можливостей Full-Stack застосунку для продажу комп'ютерних ігор	07.04.2026	Виконано
5	Реалізація інтерфейсу користувача системи по продажу комп'ютерних ігор	21.05.2026	Виконано
6	Оформлення пояснювальної записки бакалаврської роботи	10.06.2026	Виконано

Студент – дипломник

(підпис)

Михайлюк А.В.

(розшифровка підпису)

Керівник роботи

(підпис)

Дмитрик Т. Б

(розшифровка підпису)

АНОТАЦІЯ

Бакалаврська робота містить 79 сторінок, 21 рисунок, 4 таблиці, список використаних джерел із 35 найменувань.

Метою роботи є проєктування та реалізація сучасного масштабованого Full-Stack веб-застосунку для продажу цифрових комп'ютерних ігор, автентифікацією користувачів та повнофункціональною адміністративною підсистемою.

Об'єкт дослідження: процес цифрової дистрибуції комп'ютерних ігор через веб-платформу.

Предмет дослідження: методи проєктування та реалізації повностекових веб-застосунків електронної комерції з використанням Next.js та Supabase.

Результати дослідження: розроблено веб-застосунок «GameStore» на базі Next.js із використанням Supabase, що забезпечує функціонал каталогу ігор з фільтрацією та пошуком, та аналіз показників продажів.

У першому розділі проведено аналіз ринку цифрової дистрибуції відеоігор та порівняльний аналіз існуючих платформ, сформовано вимоги до системи.

У другому розділі визначено функціональні та нефункціональні вимоги й обґрунтовано вибір технологічного стеку.

У третьому розділі спроектовано архітектуру системи, розроблено UML-діаграми та ER-модель бази даних.

У четвертому розділі реалізовано повнофункціональний веб-застосунок, включаючи користувацький інтерфейс, адміністративну панель, бізнес-логіку та інтеграцію із зовнішніми сервісами.

У п'ятому розділі проведено функціональне тестування веб-застосунку.

Висновок: розроблений веб-застосунок відповідає поставленій меті, забезпечує повний функціонал інтернет-магазину цифрових ігор з підтримкою онлайн-оплати та аналізом діяльності системи.

КЛЮЧОВІ СЛОВА: ЕЛЕКТРОННА КОМЕРЦІЯ, ВЕБ-ЗАСТОСУНОК, ІНТЕРНЕТ-МАГАЗИН, NEXT.JS, FULL-STACK, SUPABASE, POSTGRESQL, АВТЕНТИФІКАЦІЯ, АДМІНІСТРАТИВНА ПАНЕЛЬ, LIQPAY.

ABSTRACT

Claude responded: The bachelor's thesis contains 79 pages, 21 figures, 4 tables, and a list of references comprising 35 sources.

The bachelor's thesis contains 88 pages, 21 figures, and a list of references comprising 35 sources.

The aim of the work is to design and implement a modern, scalable Full-Stack web application for selling digital computer games, and a fully functional administrative subsystem.

Object of research: the process of digital distribution of computer games through a web platform.

Subject of research: methods of designing and implementing full-stack e-commerce web applications using Next.js and Supabase.

Research results: the "GameStore" web application was developed based on Next.js with Supabase, providing a game catalog with filtering and search, data management through an administrative panel.

In the first chapter, an analysis of the digital video game distribution market and a comparative analysis of existing platforms were conducted, and system requirements were formulated.

In the second chapter, functional and non-functional requirements were defined, and the choice of technology stack was justified.

In the third chapter, the system architecture was designed, and UML diagrams and an ER model of the database were developed.

In the fourth chapter, a fully functional web application was implemented,.

In the fifth chapter, functional testing of the web application was conducted.

Conclusion: the developed web application meets the stated objective, providing complete online store functionality for digital games with online payment support and system activity analysis.

KEYWORDS: E-COMMERCE, WEB APPLICATION, ONLINE STORE, NEXT.JS, FULL-STACK, SUPABASE, POSTGRESQL, AUTHENTICATION, ADMINISTRATIVE PANEL, LIQPAY.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ.....	11
ВСТУП.....	12
1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗРОБКИ ІНФОРМАЦІЙНИХ СИСТЕМ ЦИФРОВОЇ ДИСТРИБУЦІЇ КОМП'ЮТЕРНИХ ІГОР.....	14
1.1. Особливості розробки веб-застосунку типу "GameStore".....	14
1.2. Мета і мотиваційні чинники розробки веб-застосунку.....	16
1.3. Аналіз існуючих досліджень і платформ онлайн-продажу комп'ютерних ігор.....	19
1.4. Функціональні можливості та технологічні обмеження існуючих систем цифрової дистрибуції.....	23
1.5. Висновки по розділу.....	28
1. АНАЛІЗ ВИМОГ ТА ВИБІР ТЕХНОЛОГІЙ.....	29
2.1. Постановка задачі та функціональні вимоги.....	29
2.2. Нефункціональні вимоги.....	30
2.3. Вибір та обґрунтування технологічного стеку.....	31
2.4. Висновки по розділу.....	33
3. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА СИСТЕМНІ ВИМОГИ ВЕБ-ЗАСТОСУНКУ ТИПУ "GAMESTORE".....	28
3.1. Вибір та обґрунтування інструментів для розробки	28
3.2. Проектування архітектури системи.....	36
3.3. Розробка структури системи.....	38
3.4. UML-моделювання.....	42
3.5. Представлення ER-моделі.....	49
3.6. Висновки по розділу	53
4. РЕАЛІЗАЦІЯ ІНТЕРНЕТ-МАГАЗИНУ ПРОДАЖУ КОМП'ЮТЕРНИХ ІГОР.....	54

					БР.ІП – 07.00.00.000 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Розробка масштабованого Full-Stack застосунку для продажу комп'ютерних ігор Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Михайлюк А.В.					6	
Перевір.		Дмитрик Т.Б.						
Реценз.		Романишин Т.Л.						
Н. Контр.		Піх М.М.						
Затверд.		Бандура В.В.				ІФНТУНГ, ІП-22-4		

4.1. Налаштування та конфігурація проєкту.....	54
4.2. Представлення роботи з базою даних.....	55
4.3.Реалізація інтерфейсу системи продажу комп'ютерних ігор.....	57
4.4 Висновки по розділу	66
5. ТЕСТУВАННЯ БЕЗ ЗАСТОСУНКУ.....	54
5.1. Тестування користувацької частини.....	73
5.2. Тестування адміністративної частини.....	74
5.3. Результати тестування.....	75
5.4. Висновки по розділу	77
ВИСНОВКИ.	78
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	80
ДОДАТКИ	
БІБЛІОГРАФІЧНА ДОВІДКА	

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

MVC – Model – View – Controller, шаблон побудови архітектурного програмного забезпечення.

CRUD – створити, прочитати, оновити, вилучити.

API – Application Programming Interface, інтерфейс програмування застосунків.

DB – база даних.

UI – User Interface, інтерфейс користувача.

UX – User Experience, досвід взаємодії.

CI – Continuous Integration, безперервна інтеграція.

CD – Continuous Delivery, безперервна доставка.

KPI – Key Performance Indicators, ключові показники ефективності

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Ринок цифрової дистрибуції відеоігор є одним із найбільш динамічних сегментів глобальної ІТ-індустрії. За оцінками аналітичних агентств, обсяг продажів цифрових ігор щорічно збільшується, а попит на зручні, захищені та функціональні веб-платформи для придбання ліцензійного програмного забезпечення постійно зростає. Сучасний споживач вимагає не лише широкого каталогу товарів, а й прозорого процесу оформлення замовлення, безпечної онлайн-оплати, зручного особистого кабінету та оперативної підтримки.

Незважаючи на наявність великих міжнародних платформ цифрового розповсюдження ігор, малий та середній бізнес у цій галузі відчуває гостру потребу у власних спеціалізованих рішеннях, адаптованих до локального ринку, підтримки вітчизняних платіжних систем (зокрема LiqPay) та україномовного інтерфейсу. Вирішення цієї проблеми є актуальним завданням, що зумовлює необхідність розробки сучасного повностекового веб-застосунку магазину комп'ютерних ігор.

Мета роботи — спроектувати та реалізувати сучасний масштабований веб-застосунок (EpicGame Shop) для продажу цифрових комп'ютерних ігор з інтеграцією платіжного шлюзу LiqPay, автентифікацією користувачів за допомогою NextAuth.js, хмарним сховищем даних Supabase (PostgreSQL) та повнофункціональною адміністративною підсистемою.

Задачі дослідження: проаналізувати предметну область електронної комерції цифрових товарів та існуючі аналоги; сформулювати функціональні та нефункціональні вимоги до системи; спроектувати архітектуру застосунку (клієнт–сервер, шар API, модель даних); реалізувати клієнтську частину на React із використанням Next.js App Router; реалізувати серверні маршрути API, валідацію вхідних даних, інтеграцію з LiqPay, електронною поштою та зворотним зв'язком через Telegram; забезпечити розмежування доступу між ролями користувача та адміністратора; описати тестування та можливості подальшого розвитку системи.

Об'єкт дослідження — процес цифрової дистрибуції комп'ютерних ігор

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

через веб-платформу.

Предмет дослідження — методи проєктування та реалізації повностекових веб-застосунків електронної комерції з використанням сучасних фреймворків і хмарних сервісів.

Методи дослідження: аналіз предметної області, об'єктно-орієнтоване та компонентне проєктування, методи REST API, реляційне моделювання даних, модульне тестування.

Практична цінність. Розроблений застосунок може бути впроваджений як повноцінний веб-магазин для продажу цифрових ігрових ліцензій з підтримкою онлайн-оплати в гривнях, управлінням замовленнями та аналітикою продажів.

Робота містить три розділи, список використаних джерел 35, та 3 додатки. Загальний обсяг пояснювальної записки — 79 сторінки.

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Особливості розробки веб-застосунку типу "GameStore"

Розробка веб-застосунків для цифрової дистрибуції комп'ютерних ігор є складним інженерним завданням, що поєднує в собі принципи електронної комерції, управління цифровими ліцензіями, безпеки платежів та забезпечення високої продуктивності при роботі з великими обсягами даних. Системи такого класу належать до категорії B2C (Business-to-Consumer) e-commerce платформ [1], де товаром виступають не фізичні об'єкти, а цифрові ліцензії на програмне забезпечення - у даному випадку, комп'ютерні ігри [2].

Ключові сутності предметної області:

Розроблювана система оперує такими основними сутностями:

- **Товар (гра)** – цифровий продукт, що має назву, опис, зображення, оригінальну та знижену ціну, набір тегів (жанр, платформа, особливості), статус активності та інші метадані. Кожна гра може мати декілька версій (стандартна, делюкс, колекційна), хоча в поточній реалізації це спрощено до однієї сутності.
- **Кошик (cart)** – тимчасове сховище вибраних користувачем товарів. У сучасних веб-застосунках стан кошика зазвичай зберігається на стороні клієнта (наприклад, у локальному сховищі браузера або в Redux з персистенцією), щоб забезпечити зручність роботи без необхідності постійної автентифікації.
- **Замовлення (order)** – фіксований набір товарів, придбаних користувачем у певний момент часу. Замовлення містить інформацію про покупця (email, ім'я, контактні дані), перелік придбаних ігор із цінами на момент покупки, підсумкову суму, статус оплати та унікальний ідентифікатор для відстеження.
- **Позиції замовлення (order items)** – деталізований перелік одиниць товару в межах одного замовлення. Зберігання назви та ціни гри безпосередньо в позиції замовлення (а не лише посилання на таблицю games) є важливим архітектурним рішенням, оскільки ціна або назва гри можуть змінитися після покупки, але історична точність замовлення має зберігатися.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

- **Користувач (user)** – особа, яка взаємодіє з системою. Користувачі поділяються на дві категорії: звичайні (мають права перегляду каталогу, створення замовлень, перегляду власної бібліотеки) та адміністратори (додаткові права на керування каталогом, користувачами, замовленнями та аналітикою).

- **Платіж (payment)** – транзакція, що здійснюється через платіжний шлюз (у даному випадку LiqPay). Платіж має власний статус, унікальний ідентифікатор від платіжної системи та пов'язаний з конкретним замовленням [15].

- **Заявка на повернення коштів (refund request)** – запит від користувача на повернення коштів за придбаний товар. Цей механізм є важливим для забезпечення прав споживачів та довіри до платформи.

Типова архітектура платформ цифрової дистрибуції:

Аналіз провідних платформ (Steam, Epic Games Store, [GOG.com](https://gog.com), Humble Bundle) дозволяє виділити типовий набір функціональних компонентів:

1. **Вітрина магазину (storefront)** – головна сторінка з каруселями акцій, рекомендованими іграми, новинками та розпродажами. Вітрина має бути динамічною, адаптивною та персоналізованою (для авторизованих користувачів).

2. **Каталог з фільтрацією та пошуком** – сторінки зі списком ігор, що підтримують фільтрацію за жанром, платформою, ціновим діапазоном, рейтингом, а також текстовий пошук з автодоповненням.

3. **Сторінка товару (product page)** – детальна інформація про гру: зображення, опис, системні вимоги, трейлер (опційно), рейтинги, відгуки, кнопка придбання.

4. **Кошик та оформлення замовлення (cart & checkout)** – проміжний етап між вибором товарів і їх оплатою. Користувач може переглянути обрані ігри, змінити кількість (хоча для цифрових товарів це зазвичай 1), застосувати промокод, ввести контактні дані та вибрати спосіб оплати.

5. **Платіжний шлюз (payment gateway)** – інтеграція із зовнішньою системою для безпечного прийому платежів. У розроблюваній системі використовується LiqPay — український платіжний сервіс, що підтримує оплату

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

банківськими картками, Apple Pay, Google Pay та іншими методами.

6. **Бібліотека покупок (library)** – список придбаних користувачем ігор. У повноцінних платформах бібліотека також надає можливість завантажити інсталятор, активувати ключі, запустити гру (через окремий клієнт).

7. **Система підтримки (support)** – механізм зворотного зв'язку, FAQ, створення заявок на повернення коштів.

8. **Адміністративна панель (admin panel)** – закрита частина системи, доступна лише адміністраторам, яка надає інструменти для керування каталогом, користувачами, замовленнями, перегляду аналітики та опрацювання заявок на повернення.

1.2. Мета і мотиваційні чинники розробки веб-застосунку

Метою даної дипломної роботи є проектування та реалізація сучасного веб-застосунку для продажу цифрових ліцензій на комп'ютерні ігри з інтеграцією платіжного шлюзу, персистентним зберіганням даних у хмарній базі даних (PostgreSQL на базі Supabase), автентифікацією користувачів із розмежуванням ролей та окремою адміністративною підсистемою для управління каталогом, користувачами, замовленнями та аналітикою.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. Провести аналіз предметної області електронної комерції у сфері цифрової дистрибуції ігор, вивчити існуючі рішення та визначити їхні сильні та слабкі сторони.

2. Сформулювати функціональні та нефункціональні вимоги до системи на основі виявлених потреб цільової аудиторії (кінцеві користувачі, адміністратори магазину) [3].

3. Спроекувати архітектуру системи за принципом клієнт-сервер, визначити компоненти (клієнтська частина, серверна логіка, API-шар, модель даних) та зв'язки між ними.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

4. Реалізувати клієнтську частину на основі бібліотеки React із використанням фреймворку Next.js у конфігурації App Router, забезпечивши адаптивний дизайн та зручний інтерфейс користувача.

5. Забезпечити розмежування доступу між ролями "користувач" та "адміністратор" за допомогою механізмів NextAuth.js та middleware.

6. Здійснити тестування реалізованої системи (модульне тестування ключових функцій) та окреслити напрями подальшого розвитку.

Мотиваційні чинники розробки

Насамперед, стрімкий розвиток цифрових технологій і глобалізація інтернет-простору сприяють переходу користувачів від фізичних носіїв до онлайн-сервісів, що забезпечують швидкий і зручний доступ до ігрового контенту. Створення власної платформи для цифрової дистрибуції комп'ютерних ігор зумовлене низкою об'єктивних передумов:

1.2.1. Зростання ринку цифрової дистрибуції відеоігор

Індустрія відеоігор є однією з найбільш швидкозростаючих у світовій економіці. За даними аналітичних агентств (Newzoo, Statista), глобальний ринок відеоігор у 2024 році перевищив 200 мільярдів доларів США [4], причому частка цифрових продажів постійно зростає і становить понад 80% від загального обсягу. Цифрова дистрибуція поступово витісняє фізичні носії завдяки зручності, миттєвому доступу до контенту та нижчим витратам на логістику.

Пандемія COVID-19 також суттєво вплинула на зростання інтересу до відеоігор [2], оскільки люди шукали нові форми дозвілля вдома. Це стимулювало додатковий попит на платформи цифрової дистрибуції, що призвело до значного збільшення кількості активних користувачів та обсягів продажів ігрового контенту по всьому світу. Багато компаній, зокрема Valve, Epic Games та GOG, зафіксували рекордні показники завантажень і доходів саме в цей період, що остаточно підтвердило цифрову дистрибуцію як домінуючу модель розповсюдження ігор і прискорило відмову галузі від фізичних носіїв на користь онлайн-платформ.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2.2. Високий поріг входу для незалежних розробників

Провідні платформи, такі як Steam або Epic Games Store, встановлюють жорсткі вимоги до розробників, які бажають публікувати свої ігри. Високі комісії (Steam бере 30% від вартості кожної проданої копії, Epic Games Store — 12%), складні процедури верифікації, обмежені можливості для маркетингу та аналітики створюють бар'єри для невеликих студій та індивідуальних розробників [2].

Створення альтернативної, більш гнучкої платформи дозволяє запропонувати рішення з нижчими комісіями та простішими процедурами публікації, що може бути привабливим для невеликих гравців ринку.

1.2.3 Демонстрація сучасних технологій

У рамках дипломної роботи важливо продемонструвати володіння актуальними технологіями та підходами до розробки програмного забезпечення: серверний рендеринг (Next.js), робота з хмарними базами даних (Supabase), типізація (TypeScript), стале керування станом (Redux Toolkit), стилізація за допомогою CSS-фреймворку (Tailwind CSS), валідація даних (Zod), інтеграція зовнішніх API, безпека автентифікації, модульне тестування (Vitest) тощо.

Кожна з цих технологій є затребуваною на ринку праці, тому їх комплексне використання підвищує практичну цінність роботи [10], [13].

1.2.4 Аналітика та управління бізнесом

Будь-яка комерційна платформа потребує інструментів аналітики для прийняття управлінських рішень. Адміністративна панель з візуалізацією ключових метрик (графіки продажів, кількість замовлень, найпопулярніші ігри) є важливою складовою успішного магазину. Наявність таких інструментів дозволяє адміністраторам оперативно реагувати на зміни в поведінці користувачів, своєчасно коригувати асортимент та цінову політику.

Розроблювана система включає дашборд з KPI та діаграмами на основі бібліотеки Recharts, що демонструє можливість не лише здійснювати продажі, але й аналізувати їх результати [20].

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

Очікувані результати

В результаті виконання дипломної роботи планується отримати:

- Повнофункціональний веб-застосунок, розгорнутий на хостингу (Netlify або аналог), доступний для тестування.
- Комплект вихідного коду з докладною документацією (README.md, коментарі в коді).
- SQL-скрипти для створення схеми бази даних у Supabase.
- Аналітичний огляд предметної області та порівняльний аналіз існуючих рішень.
- UML-діаграми, що описують архітектуру та поведінку системи.

1.3. Аналіз існуючих досліджень і платформ онлайн-продажу комп'ютерних ігор

Для визначення оптимальної функціональності, архітектурних рішень та уникнення недоліків, властивих існуючим системам, було проведено детальний аналіз провідних платформ цифрової дистрибуції комп'ютерних ігор. Основними об'єктами порівняльного аналізу обрано:

1. **Steam** (Valve Corporation) – найбільша та найстаріша платформа цифрової дистрибуції, запущена у 2003 році.
2. **GOG.com** (CD Projekt) – платформа, що спеціалізується на DRM-free іграх (без захисту від копіювання).
3. **Humble Bundle** – платформа, відома своєю моделлю "pay what you want" (плати скільки хочеш) та благодійними ініціативами.
4. **itch.io** – платформа, орієнтована на незалежних розробників та інді-ігри.

Додатково проаналізовано наукові публікації та дослідження, присвячені питанням цифрової дистрибуції, e-commerce та безпеки онлайн-платежів [6].

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

1.3.1. Аналіз платформи Steam

Steam є беззаперечним лідером ринку цифрової дистрибуції комп'ютерних ігор. Станом на 2026 рік, платформа налічує понад 120 мільйонів активних щомісячних користувачів, а каталог нараховує понад 50 000 ігор.

Сильні сторони Steam

Найбільший каталог ігор – наявність практично всіх відомих ігор, як великих блокбастерів (AAA-проекти), так і невеликих інді-ігор. Користувачі можуть знайти в Steam майже будь-яку гру, яка їх цікавить, що робить платформу універсальним місцем для придбання цифрових розваг.

Розвинена спільнота – система відгуків, форумів, груп, обговорень створює потужну соціальну складову. Користувачі можуть не лише купувати ігри, але й спілкуватися з однодумцями, обмінюватися досвідом, публікувати власний контент (наприклад, керівництва до ігор, арт, відео).

Слабкі сторони Steam

Високі комісії – Steam бере 30% комісії з кожного продажу (для великих розробників може бути знижка до 20% після досягнення певного обсягу продажів). Це є одним з найвищих показників у галузі, що викликає критику з боку розробників, особливо незалежних студій.

Складна система публікації – процес публікації гри в Steam є досить складним і включає сплату реєстраційного внеску (100 доларів США за одну гру, який повертається після досягнення певного порогу продажів), проходження верифікації, дотримання технічних та юридичних вимог. Для невеликих розробників це може бути суттєвим бар'єром [4]. Особливо гостро ця проблема відчувається в країнах з низьким рівнем доходів, де навіть відносно невеликий реєстраційний внесок може виявитися непосильним для інді-розробників-початківців.

Технологічні особливості Steam

Steam побудована на пропрієтарному програмному забезпеченні Valve. Клієнтська частина доступна для Windows, macOS, Linux, а також у вигляді мобільних додатків (iOS, Android).

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

1.3.3. Аналіз платформи GOG.com

GOG.com (раніше Good Old Games) було запущено у 2008 році польською компанією CD Projekt, відомою за серією ігор The Witcher та Cyberpunk 2077.

Сильні сторони GOG.com

DRM-free політика – головна відмінність GOG: усі придбані ігри можна завантажити у вигляді звичайних інсталяторів без будь-яких DRM-обмежень. Користувач має повний контроль над своїми придбаннями, може копіювати інсталятори на будь-які носії, встановлювати ігри без підключення до інтернету або запуску клієнта.

Адаптація класичних ігор – GOG спеціалізується на адаптації старих ігор (1990-2000-х років) для роботи на сучасних версіях Windows, macOS та Linux. Команда GOG використовує емулятори (DOSBox, ScummVM) або патчі для забезпечення сумісності, що дозволяє новим поколінням гравців познайомитися з класикою.

Слабкі сторони GOG.com

Менший каталог сучасних ігор – багато великих розробників та видавців не хочуть випускати свої ігри без DRM, тому каталог сучасних AAA-ігор на GOG є обмеженим. Наприклад, ігри від Rockstar Games, Electronic Arts, Activision Blizzard часто відсутні або з'являються із затримкою. Через це гравці, які хочуть мати доступ до найновіших блокбастерів, змушені звертатися до інших платформ. Таким чином, попри привабливу політику відсутності DRM, GOG поступово втрачає конкурентоспроможність порівняно з тими ж Steam чи Epic Games Store. Це суттєво звужує аудиторію платформи.

1.3.5. Аналіз платформи itch.io

itch.io є платформою, орієнтованою на незалежних розробників (інді), невеликі студії та окремих ентузіастів. Вона відома своєю гнучкою ціновою політикою (розробник сам встановлює мінімальну ціну, може дозволити "pay what you want" або взагалі безкоштовне завантаження), відсутністю комісії (за замовчуванням 10%, але розробник може встановити власний відсоток або

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

відмовитися від комісії на користь itch.io) та підтримкою широкого спектру форматів ігор (Windows, macOS, Linux, Android, HTML5) [4].

Сильні сторони itch.io

Максимальна гнучкість для розробників – розробники можуть самостійно визначати ціну, комісію, правила розповсюдження, додавати будь-які додаткові файли (наприклад, саундтреки, концепт-арт, джерельний код).

Відсутність жорсткого відбору – будь-хто може опублікувати гру на itch.io без попередньої модерації (хоча є система скарг для видалення неприйняттого контенту). Це робить платформу ідеальним місцем для експериментальних, нестандартних або навчальних проєктів.

Підтримка HTML5 ігор – на itch.io можна грати в HTML5-ігри безпосередньо у браузері, без завантаження. Це знижує бар'єр входу для користувачів.

Простий веб-інтерфейс – інтерфейс itch.io є легким, швидким та зручним, без зайвого навантаження.

Слабкі сторони itch.io

Низька популярність серед масового користувача – itch.io відома в основному в колах розробників та ентузіастів, але не масовому гравцю. Продажі на itch.io зазвичай значно нижчі, ніж на Steam або Epic Games Store.

Велика кількість низькоякісних ігор – через відсутність модерації, itch.io переповнена сирими, незавершеними або відверто поганими іграми. Користувачеві важко знайти якісний контент. Пошук справді вартісних проєктів перетворюється на тривалий і виснажливий процес серед сотень посередніх робіт. Це відлякує нових користувачів і підриває довіру до платформи загалом. Алгоритми рекомендацій не справляються з належним сортуванням контенту, тому навіть талановиті розробники часто залишаються непоміченими на тлі загального інформаційного шуму.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		19

1.4. Функціональні можливості та технологічні обмеження існуючих систем цифрової дистрибуції

На основі проведеного аналізу виділено ключові функціональні можливості, які мають бути реалізовані в розроблюваній системі, а також обмеження, яких слід уникати [9].

1.4.1. Функціональні вимоги до системи

1. **Перегляд головної сторінки** – користувач бачить вітрину магазину з акційними пропозиціями, мега-розпродажами, рекомендованими іграми, новинками. Головна сторінка має бути динамічною та привабливою.

2. **Перегляд каталогу** – користувач може переглядати список доступних ігор, використовувати фільтри (за жанром, платформою, ціновим діапазоном, рейтингом) та сортування (за ціною, назвою, датою виходу, популярністю). Реалізовано пагінацію для зручності навігації при великій кількості товарів.

3. **Пошук ігор** – текстовий пошук з автодоповненням, який дозволяє швидко знайти гру за назвою або ключовими словами. Пошук реалізовано як на клієнтській стороні (через API), так і у вигляді випадаючого списку в шапці сайту.

4. **Сторінка товару** – детальна інформація про гру: зображення (обкладинка, скріншоти), опис, характеристики (жанр, платформа, дата виходу, розробник, видавець), ціна (оригінальна та зі знижкою, якщо застосовано), кнопка додавання в кошик. Опційно: рейтинг, відгуки, трейлер, системні вимоги.

5. **Інформаційні сторінки** – статичні сторінки з інформацією про магазин ("Про нас"), умови використання, політику конфіденційності, контакти та підтримку. Ці сторінки мають бути доступними для всіх користувачів.

6. **Кошик** – користувач може додавати ігри в кошик, переглядати додані товари, змінювати кількість (хоча для цифрових товарів зазвичай 1), видаляти товари, переглядати підсумкову суму. Стан кошика зберігається між сесіями (редукса персистенція).

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

7. **Оформлення замовлення (checkout)** – користувач вводить контактні дані (email, ім'я, опційно телефон), може застосувати промокод (перевірка через API), переглядає підсумок замовлення та обирає спосіб оплати (LiqPay).

8. **Оплата через LiqPay** – після підтвердження замовлення користувач перенаправляється на захищену сторінку LiqPay для введення платіжних даних. Після оплати користувач повертається на сайт для перегляду результату.

9. **Зворотний зв'язок** – форма для відправки повідомлення адміністрації (запитання, скарги, пропозиції). Повідомлення надсилаються в Telegram-бот для оперативного опрацювання.

Функціонал для зареєстрованого користувача (розширює можливості гостя):

1. **Автентифікація** – вхід за допомогою email/пароля або через OAuth-провайдерів (Google, GitHub). Реєстрація нового облікового запису з підтвердженням email (опційно).

2. **Відновлення пароля** – функція "забули пароль", яка надсилає на email посилання для скидання пароля з унікальним токеном. Токени зберігаються в таблиці password_reset_tokens з обмеженим часом дії.

3. **Бібліотека покупок** – список придбаних користувачем ігор. Для кожної гри може бути передбачена можливість завантаження (якщо реалізовано) або активації ключа.

4. **Історія замовлень** – перелік усіх замовлень користувача з деталями: дата, сума, статус оплати, перелік придбаних ігор.

5. **Збережені ігри (wishlist)** – користувач може додавати ігри в список бажаних, щоб не втратити їх з поля зору та отримувати сповіщення про знижки (опційно).

6. **Заявки на повернення коштів** – користувач може створити заявку на повернення коштів за придбану гру, вказавши причину. Адміністратор розглядає заявку та приймає рішення.

7. **Нещодавно переглянуті** – відстеження ігор, які користувач переглядав, для персоналізованих рекомендацій.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

Функціонал для адміністратора (розширює можливості користувача):

1. **Адмін-панель** – закрита частина системи, доступна лише користувачам з прапорцем `is_admin = true`. Доступ до маршрутів `/admin/*` захищено `middleware`.

2. **Дашборд (Dashboard)** – головна сторінка адмін-панелі з ключовими показниками (KPI): кількість замовлень за період, загальна виручка, кількість зареєстрованих користувачів, кількість активних ігор. Візуалізація за допомогою діаграм (Recharts).

3. **Керування іграми (CRUD)** – адміністратор може:

- Переглядати список усіх ігор (активних та видалених)
- Додавати нову гру (заповнення форми: назва, опис, ціна, зображення, теги)
- Редагувати існуючу гру (зміна ціни, оновлення опису, завантаження нового зображення)
- Виконувати м'яке видалення (встановлювати `is_active = false` без фізичного видалення з БД)

4. **Керування користувачами** – адміністратор може:

- Переглядати список усіх зареєстрованих користувачів (email, дата реєстрації, статус)
- Блокувати/розблокувати користувачів (заблокований користувач не може автентифікуватися)
- Призначати або знімати права адміністратора (опційно)

5. **Керування замовленнями** – адміністратор може:

- Переглядати всі замовлення (незалежно від користувача) з фільтрацією за статусом та датою
- Змінювати статус замовлення (`pending` → `paid` → `cancelled`)
- Переглядати деталі замовлення (склад, суми, дані покупця)

6. **Аналітика** – сторінка з розширеною аналітикою, де адміністратор може:

- Обрати період (день, тиждень, місяць, довільний інтервал)

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

- Переглянути графік продажів за обраний період
- Переглянути найпопулярніші ігри (топ-10 за кількістю продажів або виручкою)
- Експортувати звіти (опційно)

1.4.2. Нефункціональні вимоги

- Всі секрети (API-ключі, паролі, токени) зберігаються виключно в змінних середовищах (файл .env.local), який не потрапляє в репозиторій (додано в .gitignore) [14].
- SUPABASE_SERVICE_ROLE_KEY використовується лише на серверній стороні (API-маршрути, скрипти) і ніколи не передається клієнту.
- Паролі користувачів зберігаються в БД у вигляді хешів (bcrypt), ніколи у відкритому вигляді.
- Вхідні дані API-запитів валідуються за допомогою Zod-схем; некоректні дані відхиляються з відповідним HTTP-кодом (400, 422).
- Адмін-маршрути захищені middleware, яке перевіряє наявність isAdmin в JWT-токені.
- Логування (lib/logger.ts) не розкриває внутрішні деталі (наприклад, повний стек помилок) кінцевому користувачу в API-відповідях.

Надійність:

- Валідація тіл запитів на рівні API запобігає потраплянню некоректних даних у БД.
- З'єднання з базою даних через Supabase є стійким до короткочасних перерв (автоматичні повторні спроби).

Продуктивність:

- Використання Next.js Server Components дозволяє частково рендерити сторінки на сервері, зменшуючи навантаження на клієнта [5].

Підтримуваність:

- Модульна структура проєкту: app/ (маршрутизація), components/ (UI-компоненти), lib/ (утиліти, API-клієнти), types/ (TypeScript-типи).

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

- Використання TypeScript забезпечує статичну типізацію, що полегшує рефакторинг та виявлення помилок на етапі компіляції.

- Документація у вигляді README.md та коментарів у коді (особливо для складних частин).

Тестування:

- Модульні тести (Vitest) для ключових чистих функцій:

Розгортання:

- Змінні середовища конфігуруються через .env.local (для локального запуску) або через панель хостингу (Netlify, Vercel).

- Збірка продакшен-версії: npm run build.

- Запуск: npm run start (після збірки) або npm run dev (режим розробки з hot-reload).

1.5 Висновки по розділу:

Проведено аналіз ринку цифрової дистрибуції відеоігор, який демонструє стійке зростання як за обсягом продажів, так і за кількістю користувачів цифрових платформ. Розглянуто провідні існуючі платформи — Steam, Epic Games Store, GOG та інші — з метою виявлення їхніх сильних сторін, недоліків та особливостей побудови користувацького досвіду. Встановлено, що більшість існуючих рішень орієнтовані на глобальний ринок і не враховують специфіку локальних платіжних систем та способів доставки цифрового контенту. Визначено нефункціональні вимоги до системи, що охоплюють продуктивність, масштабованість, безпеку даних користувачів, адаптивність інтерфейсу для мобільних пристроїв та доступність застосунку. Виявлені недоліки конкурентних рішень стали основою для формування ключових функціональних вимог до розроблюваної системи. Отримані результати аналізу забезпечують теоретичне підґрунтя для прийняття обґрунтованих архітектурних та технологічних рішень у наступних розділах роботи.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА СИСТЕМНІ ВИМОГИ ВЕБ-ЗАСТОСУНКУ ТИПУ "GAMESTORE"

2.1. Вибір та обґрунтування інструментів для розробки

Правильний вибір технологічного стеку є критично важливим етапом розробки будь-якого програмного забезпечення. Від нього залежить не тільки швидкість та зручність розробки, але й продуктивність, безпека, масштабованість та підтримуваність кінцевої системи [3]. При виборі інструментів для розробки веб-застосунку "GameStore" враховувалися такі критерії:

- **Сучасність технології** – використання актуальних версій фреймворків та бібліотек, які активно підтримуються та мають велику спільноту.
- **Продуктивність** – здатність системи швидко обробляти запити та ефективно працювати під навантаженням.
- **Безпека** – наявність вбудованих механізмів захисту від поширених веб-уразливостей.
- **Масштабованість** – можливість нарощування функціональності та продуктивності в майбутньому.
- **Сумісність** – здатність різних інструментів ефективно працювати разом.

2.1.1. Фреймворк Next.js

Next.js є фреймворком на основі бібліотеки React, який надає розширені можливості для створення серверних веб-застосунків. Він був розроблений компанією Vercel і на сьогодні є одним з найпопулярніших рішень для створення сучасних веб-проектів [5], [11].

Основні переваги Next.js, які зумовили його вибір:

App Router (маршрутизація на основі папок), Server Components та Client Components, API Route Handlers. Однією з ключових переваг Next.js є можливість

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

створювати API-ендпоінти безпосередньо в межах одного проєкту. *Middleware*. Механізм *middleware* дозволяє перехоплювати запити до маршрутів до їх обробки основними обробниками, *TypeScript* підтримка, *Оптимізація зображень*. Вбудована підтримка *CSS* та *Tailwind*. Next.js легко інтегрується з Tailwind CSS, що було обрано для стилізації проєкту.

Версія: у проєкті використовується Next.js 16.1.1 (актуальна стабільна версія на момент розробки).

Недоліки Next.js (які враховано при проектуванні):

- Деякі функції App Router все ще перебувають у стані активної розробки, тому окремі підходи можуть змінюватися.
- Для складних серверних обчислень може знадобитися окремий бекенд, але для потреб "GameStore" можливостей Next.js достатньо.

2.1.2. Хмарна база даних Supabase (PostgreSQL)

Для зберігання даних проєкту було обрано Supabase – хмарну платформу з відкритим кодом, яка надає базу даних PostgreSQL, автентифікацію, сховище файлів та інші сервіси [7].

Обґрунтування вибору Supabase:

PostgreSQL. Supabase використовує повноцінну реляційну базу даних PostgreSQL, яка є однією з найпотужніших та найнадійніших СКБД у світі. PostgreSQL підтримує складні запити, транзакції, зовнішні ключі, індекси, тригери, що дозволяє реалізувати складну бізнес-логіку безпосередньо на рівні бази даних. Для "GameStore" це важливо, оскільки модель даних включає зв'язки між таблицями (замовлення ↔ позиції замовлення, користувачі ↔ замовлення) [9].

Хмарне рішення. Використання хмарної бази даних позбавляє від необхідності адмініструвати власний сервер БД, виконувати резервне копіювання, налаштовувати реплікацію. Supabase автоматично створює резервні копії, забезпечує високу доступність та масштабування [11].

Інтеграція з NextAuth. Supabase може використовуватися як адаптер для NextAuth.js для зберігання сесій та користувачів [16].

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

Доступність. Supabase має безкоштовний тарифний план, який надає до 500 МБ бази даних, що цілком достатньо для навчального проєкту [7].

Схема бази даних:

У Supabase створено наступні таблиці (детальний опис наведено в підрозділі 2.5):

- users – користувачі системи (розширює стандартну таблицю auth.users з Supabase).
- games – каталог ігор.
- orders – замовлення.
- order_items – позиції замовлень.
- password_reset_tokens – токени для скидання пароля.
- refund_requests – заявки на повернення коштів.
- promo_codes – промокоди для знижок.
- users — користувачі системи
- **games** — каталог ігор із повною інформацією про продукт, включаючи опис, ціну, жанр, платформу та медіаматеріали.
- **orders** — замовлення користувачів із зазначенням статусу.

order_items — позиції замовлень, що зберігають детальну інформацію про кожен товар у складі конкретного замовлення, зокрема посилання на гру, її ціну на момент покупки та кількість одиниць.

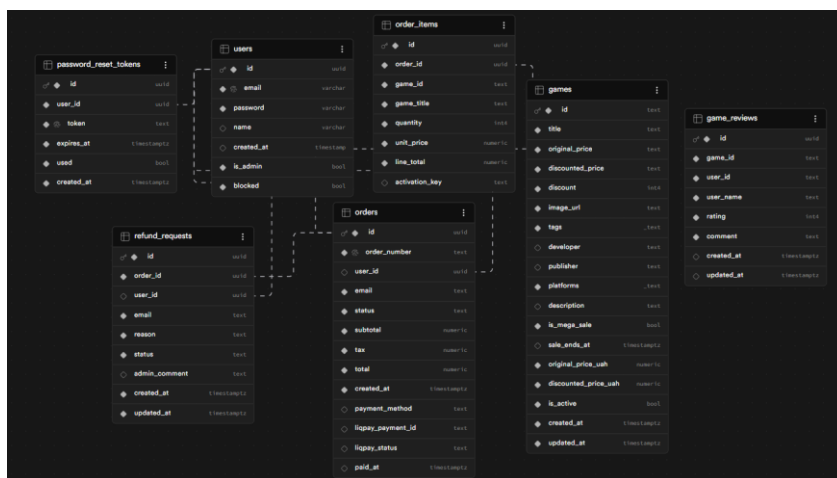


Рисунок. 2.1 – Логічна модель даних модулів кошика, замовлення та платіжних транзакцій

					БР.ІП - 07.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		27

Таблиця **password_reset_tokens** забезпечує безпечний механізм відновлення доступу, зберігаючи тимчасові токени з обмеженим терміном дії. Логічну модель даних наведено на рисунку 2.1

2.1.3. Інструмент автентифікації NextAuth.js

Для забезпечення автентифікації та управління сесіями користувачів було обрано NextAuth.js (версія 4.24.13). Це бібліотека, спеціально розроблена для Next.js, яка забезпечує гнучку та безпечну автентифікацію з підтримкою багатьох провайдерів [16].

Обґрунтування вибору NextAuth.js:

Інтеграція з Next.js. NextAuth.js є "рідним" рішенням для Next.js, воно безпосередньо інтегрується з App Router, API-маршрутами та middleware. Це забезпечує просту конфігурацію та високу продуктивність [8].

Підтримка множинних провайдерів. NextAuth.js підтримує понад 50 різних провайдерів автентифікації (Google, GitHub, Facebook, Twitter, Apple, LinkedIn та інші), а також кастомні провайдери (зокрема, Credentials для входу за email/паролем). У "GameStore" реалізовано:

- **Credentials Provider** – класичний вхід за email та паролем (з хешуванням bcrypt).
- **Google Provider** – вхід через обліковий запис Google (OAuth 2.0).
- **GitHub Provider** – вхід через обліковий запис GitHub (OAuth 2.0).

2.1.4. Мова програмування TypeScript

TypeScript є статично типізованою надбудовою над JavaScript, розробленою компанією Microsoft. Він додає до JavaScript можливість визначати типи змінних, параметрів функцій, об'єктів та класів, що дозволяє виявляти помилки на етапі компіляції, а не під час виконання [9].

Обґрунтування вибору TypeScript:

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

Статична типізація. JavaScript є динамічно типізованою мовою, що робить його гнучким, але водночас збільшує ризик помилок, пов'язаних з невідповідністю типів. TypeScript вирішує цю проблему: помилка типу (наприклад, передача числа у функцію, яка очікує рядок) буде виявлена ще під час написання коду або на етапі компіляції [10].

Покращена підтримка IDE. TypeScript надає розширену інформацію для інтегрованих середовищ розробки (автодоповнення, навігація по коду, рефакторинг). Це значно прискорює розробку та зменшує кількість помилок.

Використання TypeScript у проєкті "GameStore":

У проєкті TypeScript використовується для:

- **Типізації сутностей** – визначено типи для гри (types/game.ts), замовлення, користувача, заявки на повернення.
- **Типізації API-відповідей** – кожен API-ендпоінт має чітко визначений тип відповіді (наприклад, GameCatalogResponse).
- **Типізації пропсів компонентів** – кожен React-компонент має інтерфейс пропсів, що запобігає помилкам при передачі даних.
- **Розширення типів NextAuth** – у файлі types/next-auth.d.ts розширено стандартні типи сесії та JWT для додавання поля isAdmin.
- **Валідації за допомогою Zod** – Zod (бібліотека для валідації даних) використовується разом з TypeScript для автоматичного виведення типів.

2.1.5. Платіжний шлюз LiqPay

LiqPay є українським платіжним сервісом, заснованим у 2014 року як дочірня компанія ПриватБанку. LiqPay дозволяє приймати онлайн-платежі банківськими картками (Visa, MasterCard), через Apple Pay, Google Pay, Privat24, а також через термінали самообслуговування [13].

Обґрунтування вибору LiqPay:

Локальна орієнтація. Оскільки застосунок розробляється в Україні, використання українського платіжного сервісу є логічним вибором. LiqPay

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

підтримує розрахунки в гривнях (UAH), що дозволяє уникнути конвертації валют та пов'язаних з цим комісій [15].

Простота інтеграції. LiqPay надає просте REST API та детальну документацію. Інтеграція може бути виконана як у вигляді перенаправлення користувача на сторінку LiqPay (Hosted Page), так і через вбудовану платіжну форму (Iframe). У "GameStore" використовується перенаправлення, що є найпростішим та найбезпечнішим методом.

2.1.6. Середовище розробки Visual Studio Code

Visual Studio Code (VS Code) є безкоштовним, легковажним та потужним редактором коду з відкритим кодом, розробленим компанією Microsoft. Він був обраний як основне середовище для розробки проєкту "GameStore".

Переваги VS Code, що вплинули на вибір:

Легкість та швидкодія. На відміну від важких IDE (наприклад, WebStorm, IntelliJ IDEA), VS Code споживає менше ресурсів та швидше запускається, що важливо при роботі на різних комп'ютерах [14].

Розширюваність. VS Code має величезний маркетплейс з тисячами розширень для будь-яких потреб. Для роботи з проєктом встановлено наступні розширення:

- **TypeScript/JavaScript** – вбудована підтримка.
- **ESLint** – перевірка стилю коду.
- **Prettier** – автоматичне форматування коду.
- **Tailwind CSS IntelliSense** – автодоповнення для Tailwind CSS класів.
- **GitLens** – розширена робота з Git.
- **Thunder Client** – тестування API-ендпоінтів.

Інтеграція з Git. VS Code має вбудовану підтримку систем контролю версій, що дозволяє виконувати основні Git-операції (commit, push, pull, перегляд змін) без використання терміналу. Це значно спрощує процес розробки, особливо для початківців, оскільки всі ключові дії доступні через зручний графічний інтерфейс.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

2.1.7. Програмні вимоги

Для розробки, запуску та експлуатації веб-застосунку "GameStore" необхідні наступні програмні та апаратні засоби. Правильний вибір інструментів та технологій безпосередньо впливає на продуктивність, надійність і масштабованість системи. Нижче наведено перелік основних вимог, які забезпечують стабільну роботу застосунку на всіх етапах його життєвого циклу. Дотримання цих вимог є обов'язковою умовою для успішного розгортання та подальшої підтримки проєкту.

Таблиця 2.1

Програмні вимоги для розробки

Компонент	Мінімальна версія	Підтримка
Node.js	18.x або вище	JavaScript runtime
Npm	9.x або вище	Менеджер пакетів
Git	2.x	Система контролю версій
Visual Studio Code	1.80+	Рекомендоване середовище розробки
Браузер	Сучасний (Chrome, Firefox, Edge)	Для тестування UI
Supabase обліковий запис	-	Безкоштовний тарифний план

Залежності проєкту (package.json):

Основні залежності (dependencies):

Файл package.json визначає всі зовнішні бібліотеки та інструменти, які використовуються у проєкті. Залежності поділяються на основні (dependencies), необхідні для роботи застосунку у production-середовищі, та залежності розробки (devDependencies), які використовуються лише під час розробки та збірки проєкту. Перелік основних залежностей наведено на рисунку 2.2 **next** — фреймворк для React-застосунків із підтримкою серверного рендерингу, файлової маршрутизації та оптимізації продуктивності.

```

{
  "name": "gamestore",
  "version": "0.1.0",
  "private": true,

  "scripts": {
    "clean": "node -e \"require('fs').rmSync('.next',{recursive:true,force:true});\"",
    "dev": "next dev --webpack",
    "dev:turbo": "next dev --turbopack",
    "build": "next build",
    "build:webpack": "next build --webpack",
    "start": "next start",
    "lint": "eslint",
    "test": "vitest run",
    "seed:admin": "tsx scripts/seed-admin.ts",
    "seed:games": "tsx scripts/seed-games.ts"
  },
}

```

Рисунок 2.2 – Перелік основних залежностей та інструментів розробки Next.js застосунку

2.2. Проектування архітектури системи

Архітектура програмного забезпечення визначає основні компоненти системи, їхні обов'язки та способи взаємодії. Для веб-застосунку "GameStore" обрано клієнт-серверну архітектуру з використанням сучасних підходів: серверний рендеринг (SSR), статична генерація (SSG), клієнтський рендеринг (CSR) та API-маршрути в межах одного застосунку [5].

2.2.1. Загальна архітектурна схема

Система складається з наступних основних компонентів:

1. Клієнтська частина (Frontend):

- Браузер користувача.
- React-компоненти, які формують інтерфейс.
- Redux-сховище для керування станом кошика (з redux-persist для збереження між сесіями).
- Next.js Client Components для інтерактивних елементів.
- API Route Handlers (app/api/*) для обробки HTTP-запитів.

2. Серверна частина (Next.js Server):

- Server Components для рендерингу сторінок на сервері.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

- API Route Handlers (app/api/*) для обробки HTTP-запитів.
 - NextAuth.js для керування автентифікацією та сесіями.
3. **Шар даних (Data Layer):**
- Supabase (PostgreSQL) як основне сховище даних.
 - Клієнти Supabase для взаємодії з БД (з різними рівнями доступу).

4. **Зовнішні сервіси (External Services):**

- **LiqPay** – платіжний шлюз для прийому онлайн-платежів.
- **Resend** – сервіс для відправки транзакційних email (чеки, відновлення пароля).
- **Telegram Bot API** – для надсилання повідомлень з форми зворотного зв'язку.
- **Google OAuth / GitHub OAuth** – для автентифікації через сторонні провайдери.
- NextAuth.js для керування автентифікацією та сесіями.
- API Route Handlers (app/api/*) для обробки HTTP-запитів.
- Redux-сховище для керування станом кошика (з redux-persist для збереження між сесіями).

2.2.2. Діаграма архітектури

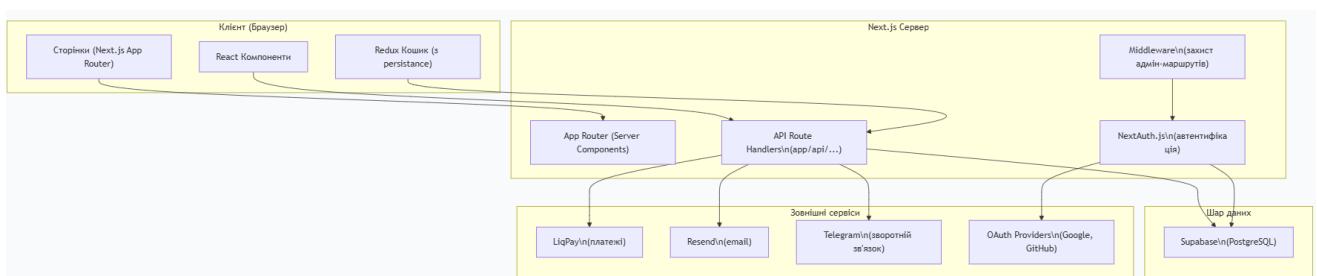


Рисунок 2.3 – Архітектура клієнт-серверної взаємодії інтернет-магазину

2.3. Розробка структури системи

Структура файлів та папок проєкту відображає його архітектуру та

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

забезпечує зручність навігації та підтримки коду. У цьому підрозділі наведено організацію проєкту "GameStore".

2.3.1. Коренева структура

Коренева структура веб-застосунку "GameStore" визначає загальну організацію файлів і каталогів проєкту. Вона є основою архітектури застосунку та забезпечує логічне розмежування між різними компонентами системи. Правильно побудована коренева структура спрощує навігацію по проєкту, полегшує командну розробку та підтримку коду. Структуру каталогів зображено на рисунку 2.4. Кожна директорія виконує чітко визначену роль і містить лише ті файли, які відповідають її призначенню. Така організація є запорукою довгострокової підтримуваності проєкту та його успішного масштабування в майбутньому.

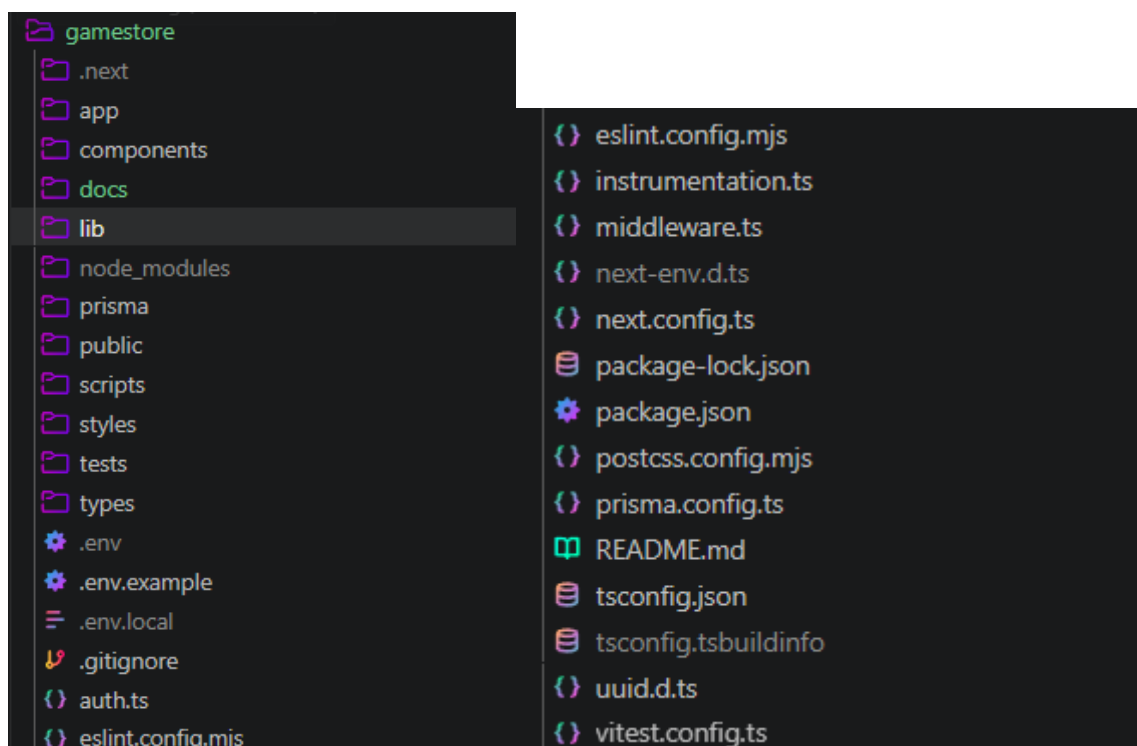


Рисунок 2.4 – Структура каталогів та файлів проєкту Gamestore

2.3.2. Структура app/ (маршрутизація)

Папка app/ реалізує маршрутизацію Next.js App Router. Кожна підпапка

відповідає сегменту URL. Структуру маршрутів показано на рисунку 2.5. Папка `app/` є центральним елементом архітектури проєкту та реалізує файлову маршрутизацію відповідно до концепції App Router, введеної у Next.js 13. Кожна підпапка всередині `app/` відповідає окремому сегменту URL-адреси, а наявність файлу `page.tsx` у папці робить цей маршрут доступним для користувача. Файли з особливими назвами виконують визначені фреймворком функції: `layout.tsx` задає спільну обгортку для групи сторінок, `loading.tsx` відображає індикатор завантаження під час отримання даних. Файл `error.tsx` відповідає за обробку помилок на рівні окремого маршруту, забезпечуючи коректне відображення повідомлень про збої без перезавантаження всієї сторінки. Окрім цього, файл `not-found.tsx` дозволяє визначити користувацьку сторінку для неіснуючих маршрутів, що покращує загальний досвід користувача.

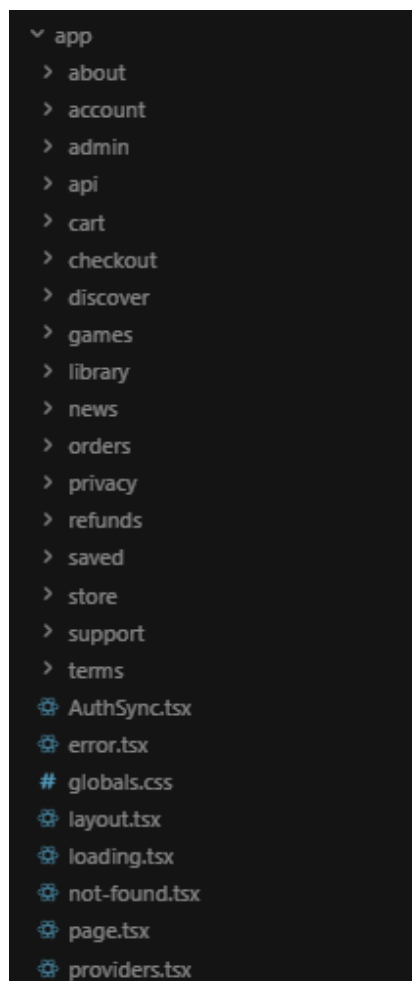


Рисунок 2.5 – Структура маршрутів Next.js App Router у каталозі `app/`

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

На кореновому рівні папки `app/` розміщено такі спеціальні файли: **AuthSync.tsx** — компонент синхронізації стану автентифікації між Supabase та клієнтською частиною застосунку; **error.tsx** — глобальний обробник помилок; **globals.css** — глобальні стилі застосунку; **layout.tsx** — кореневий макет, що обгортає всі сторінки; **loading.tsx** — глобальний індикатор завантаження; **not-found.tsx** — сторінка 404; **page.tsx** — головна сторінка застосунку; **providers.tsx** — компонент, що об'єднує всіх React-провайдерів контексту (наприклад, теми, стану автентифікації тощо). Структуру API-маршрутів наведено на рисунку 2.6.

2.3.3. Структура `app/api/` (API-маршрути)

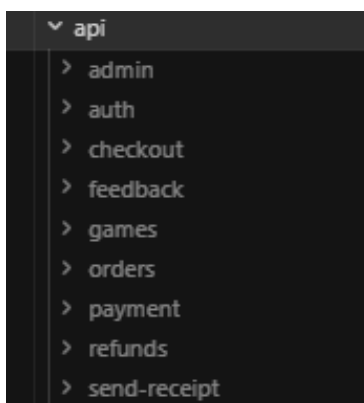


Рисунок 2.6 – Структура API-маршрутів Next.js у каталозі `app/api/`

2.3.4. Структура `components/`

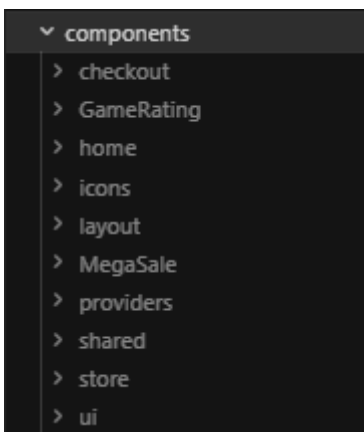


Рисунок 2.7 – Структура каталогу React-компонентів проєкту Gamestore

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

Папка components/ містить усі повторно використовувані React-компоненти проєкту, згруповані за функціональним призначенням. **checkout** — компоненти для сторінки. Структуру бібліотек наведено на рисунку 2.8.

2.3.5. Структура lib/ (бібліотеки та утиліти)

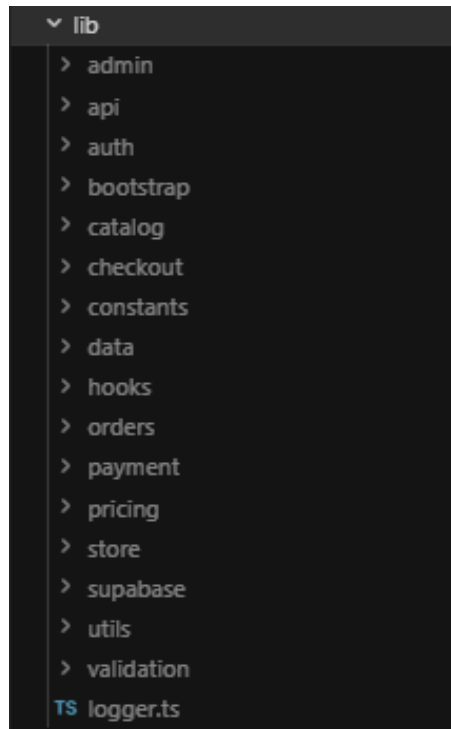


Рисунок 2.8 – Структура каталогу бібліотек та допоміжних модулів lib/

2.4. UML-моделювання

Unified Modeling Language (UML) є стандартною мовою для візуалізації, специфікації, конструювання та документування програмних систем. У цьому розділі представлено основні UML-діаграми, що описують архітектуру та поведінку системи "GameStore" [3].

2.4.1. Представлення випадків використання (Use Case Diagram)

Діаграма випадків використання (Use Case Diagram) відображає функціональні

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

вимоги до системи з точки зору її зовнішніх акторів (користувачів та зовнішніх дій). Модель ролей та інтеграція з сервісами наведені на рисунку 2.9

Актори системи:

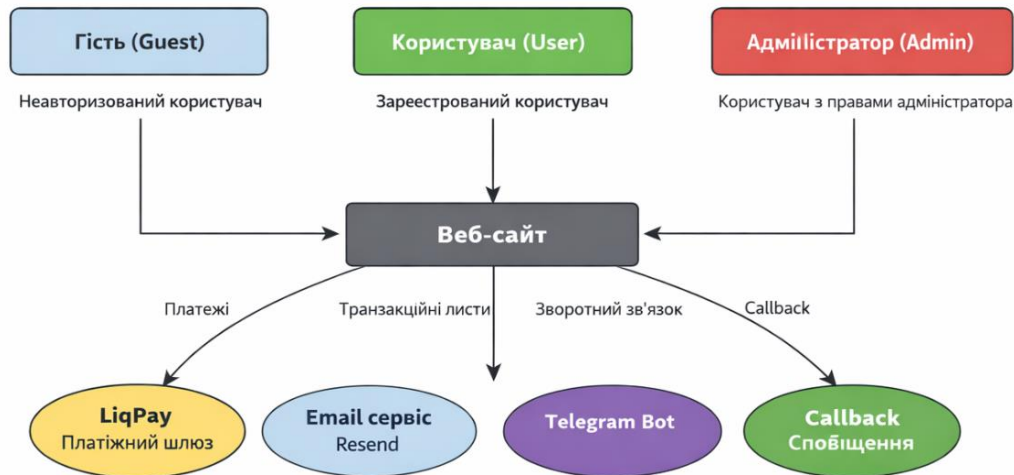


Рисунок 2.9 – Модель ролей користувачів та інтеграція з платіжними та комунікаційними сервісами

2.4.2. Представлення класів (Class Diagram)

Діаграма класів (Class Diagram) відображає статичну структуру системи: класи, їхні атрибути, методи та зв'язки між ними. У контексті веб-застосунку "GameStore" діаграма класів описує основні сутності предметної області [12].

Основні класи системи:

1. Клас **User** описує користувачів системи, їхні дані, ролі та статуси доступу.
2. Клас **Game** представляє ігровий контент із цінами, описом і характеристиками.
3. Клас **Order** разом із **RefundRequest** забезпечує процес покупки та повернення коштів, а сервіси реалізують відповідну бізнес-логіку. Діаграму класів зображено на рисунку 2.10

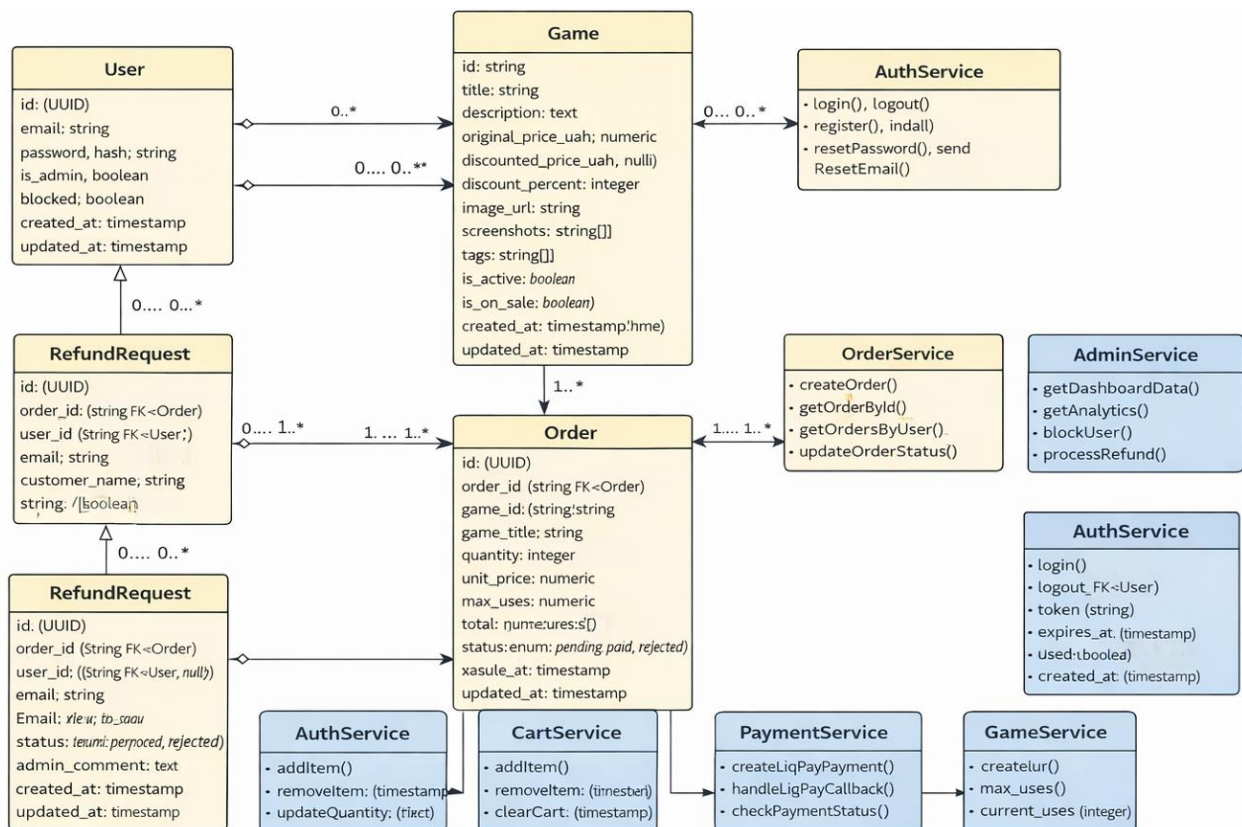


Рисунок 2.10 – Діаграма класів основних сутностей та сервісів системи Gamestore

Методи класів (для ключових сервісів):

Хоча в TypeScript-кодi класи не використовуються для сутностей (замість цього використовуються інтерфейси та функції), можна виділити логічні "сервісні" класи:

- AuthService: login(), logout(), register(), resetPassword(), sendResetEmail
- CartService: addItem(), removeItem(), updateQuantity(), clearCart(), apply PromoC
- ode()
- OrderService: createOrder(), calculateTotal(), processPayment()
- OrderService: createOrder(), calculateTotal(), processPayment()
- UserService: getUserProfile(), updateProfile(), changePassword()
- AuthService: login(), logout(), register(), resetPassword(), sendResetEmail
- CartService: addItem(), removeItem(), updateQuantity(), clearCart(), apply PromoCode()

2.4.3. Представлення послідовності (Sequence Diagram)

Діаграма послідовності (Sequence Diagram) відображає взаємодію об'єктів у часі при виконанні певного сценарію. Вона показує, які повідомлення передаються між об'єктами та в якому порядку.

Сценарій: Оформлення замовлення та оплата через LiqPay

Цей сценарій є найбільш критичним для системи, оскільки включає взаємодію з зовнішнім платіжним шлюзом [15].

Актори та об'єкти:

- Користувач – кінцевий користувач
- Браузер – веб-браузер клієнта
- Next.js Server – сервер застосунку
- Supabase – база даних
- LiqPay – платіжний шлюз
- Resend – email-сервіс

Послідовність взаємодії починається з того, що користувач ініціює оформлення замовлення у браузері, після чого запит передається на сервер. Сервер обробляє дані замовлення, зберігає їх у базі даних та формує запит до платіжного шлюзу для проведення оплати. Після успішної транзакції система отримує підтвердження, оновлює статус замовлення та надсилає користувачу повідомлення на електронну пошту.

У разі помилки під час оплати користувач отримує відповідне повідомлення, а замовлення може бути позначене як неуспішне або очікуване.

Кроки сценарію:

Діаграма послідовності відображає повний процес обробки платежу через LiqPay — від натискання кнопки «Сплатити» користувачем до успішного або неуспішного завершення транзакції з відповідним оновленням статусу замовлення у Supabase та надсиланням квитанції через Resend. На рисунку 2.11 задіяно п'ять учасників — Користувач, Браузер, Next.js Server, Supabase та LiqPay — і чітко розмежовано два сценарії.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

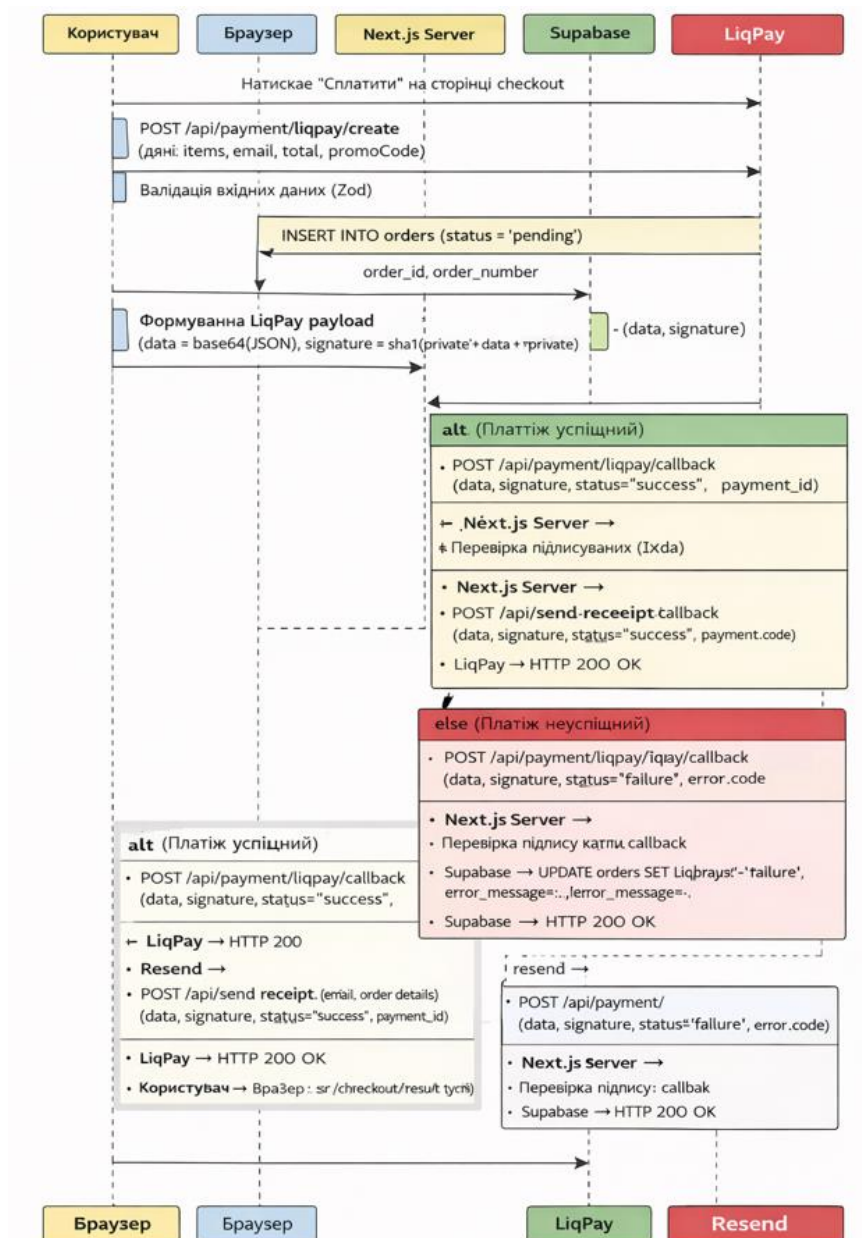


Рисунок 2.11 – Діаграма послідовності процесу оплати через платіжний шлюз

Опис сценарію у вигляді нумерованого списку для рисунка:

1. Користувач натискає кнопку "Сплатити" на сторінці оформлення замовлення.
2. Браузер надсилає POST-запит на `api/payment/liqpay/create` з даними замовлення.
3. Сервер валідує отримані дані за допомогою `Zod`.
4. Сервер створює запис замовлення в `Supabase` зі статусом `pending`.

5. Сервер формує підписаний payload для LiqPay (base64-кодований JSON та SHA1-підпис).

6. Сервер повертає клієнту data та signature.

7. Браузер надсилає форму на LiqPay.

8. LiqPay відображає платіжну форму користувачу.

9. Користувач вводить дані картки та підтверджує платіж.

10. LiqPay обробляє платіж.

11. **Якщо платіж успішний:**

- LiqPay надсилає callback-сповіщення на сервер.
- Сервер перевіряє підпис callback.
- Сервер оновлює статус замовлення в Supabase на paid.
- Сервер надсилає email-чек через Resend.
- Користувач перенаправляється на сторінку успішної оплати.

12. **Якщо платіж неуспішний:**

- LiqPay надсилає callback з помилкою.
- Сервер оновлює статус замовлення на failed.
- Користувач перенаправляється на сторінку з повідомленням про

помилку.

2.4.4. Представлення діяльності (Activity Diagram)

Діаграма діяльності (Activity Diagram) відображає потік управління від однієї дії до іншої. Вона подібна до блок-схеми, але підтримує паралельне виконання та розгалуження.

Сценарій: Процес оформлення замовлення

Ця діаграма описує дії користувача та системи від моменту додавання гри в кошик до завершення оплати.

Основні елементи діаграми діяльності:

- **Початковий вузол** – чорне коло, позначає початок процесу.
- **Кінцевий вузол** – чорне коло з обідком, позначає завершення.

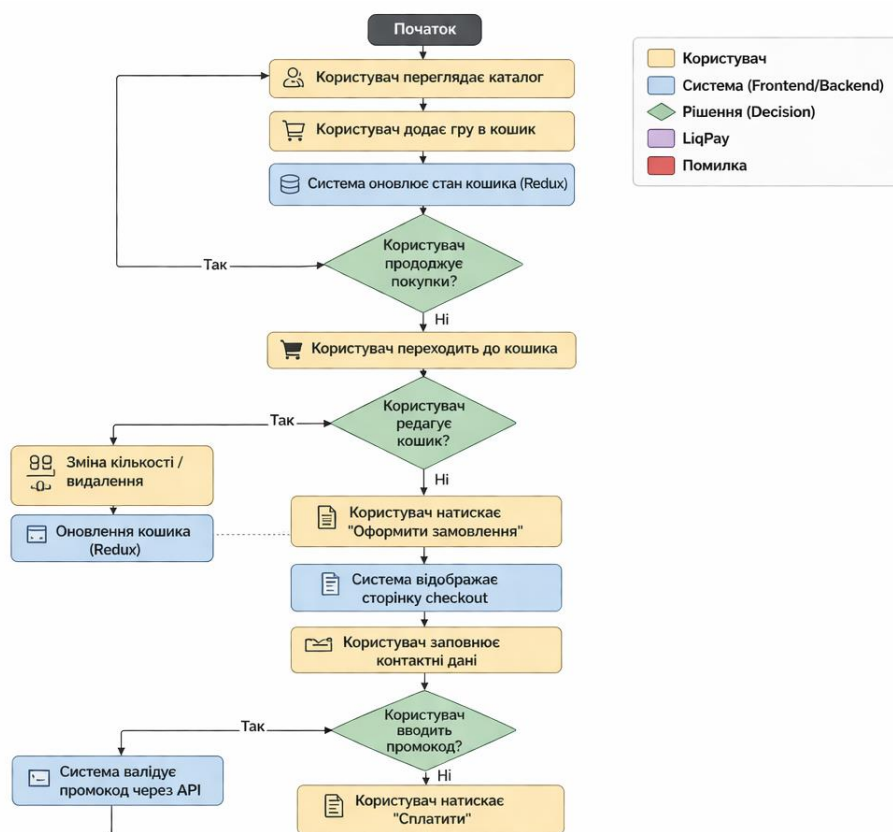
					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

- **Дія (Action)** – прямокутник із заокругленими кутами, позначає виконання певної операції.

- **Розгалуження (Decision)** – ромб, позначає умовний перехід (так/ні).

- **Потік (Flow)** – стрілка, що показує перехід між діями.

Діаграма діяльності дозволяє наочно відобразити логіку роботи системи та взаємодію між її компонентами. Вона допомагає виявити можливі помилки або зайві кроки в процесі ще на етапі проєктування. Також її зручно використовувати для пояснення роботи системи іншим розробникам або замовникам. Опис діаграми діяльності зображено на рисунку 2.12



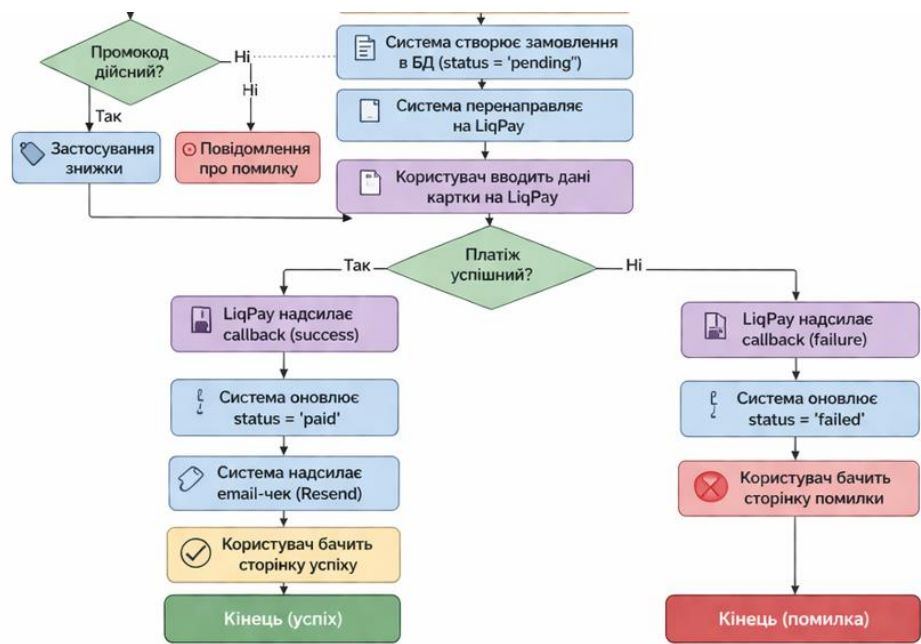


Рисунок 2.12 – Діаграма активності процесу оформлення замовлення від перегляду каталогу до завершення оплати

2.5. Представлення ER-моделі

ER-модель (Entity-Relationship model) описує структуру бази даних системи: сутності, їхні атрибути та зв'язки між ними. Для веб-застосунку "GameStore" використовується реляційна база даних PostgreSQL, керована через Supabase.

2.5.1. Перелік сутностей та їхніх атрибутів

База даних застосунку складається з дев'яти основних сутностей, кожна з яких відповідає окремій таблиці у Supabase (PostgreSQL). Сутність users зберігає профільні дані користувачів, їхню роль у системі та розширює стандартну таблицю автентифікації auth.users структуру якої показано на рисунку 2.13. Сутність games містить повну інформацію про ігри каталогу — назву, опис, поточну та стару ціну, обкладинку, скріншоти, жанр, платформу, рік випуску, рейтинг, системні вимоги та ознаку наявності в продажу яка наведена на рисунку 2.14.

Сутність users (користувачі)

users	
id	uuid
email	varchar
password	varchar
name	varchar
created_at	timestamp
is_admin	bool
blocked	bool

Рисунок 2.13 – Структура таблиці користувачів (users) бази даних Gamestore

Таблиця **games** є центральним елементом каталогу платформи та містить такі поля: **id** (text, primary key) — унікальний ідентифікатор гри; **title** (text) — назва гри; **original_price** (text) — початкова ціна; **discounted_price** (text) — ціна зі знижкою; **discount** (int4) — розмір знижки у відсотках; **image_url** (text) — посилання на обкладинку гри; **tags** (_text) — масив тегів для категоризації; **developer** (text) — розробник гри; **publisher** (text) — видавець гри; **platforms** (_text) — масив підтримуваних платформ; **description** (text) — опис гри; **is_mega_sale** (bool) — ознака участі гри в розпродажі MegaSale; **sale_ends_at** (timestampz) — дата та час завершення акції; **original_price_uah** (numeric) — початкова ціна в гривнях; **discounted_price_uah** (numeric) — знижена ціна в гривнях; **is_active** (bool) — ознака активності запису в каталозі; **created_at** (timestampz) — дата створення запису; **updated_at** (timestampz) — дата останнього оновлення запису. Структуру таблиці **games** бази даних Gamestore наведено на рисунку 2.14.

Сутність games (ігри)

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

games	
id	text
title	text
original_price	text
discounted_price	text
discount	int4
image_url	text
tags	_text
developer	text
publisher	text
platforms	_text
description	text
is_mega_sale	bool
sale_ends_at	timestampz
original_price_uah	numeric
discounted_price_uah	numeric
is_active	bool
created_at	timestampz
updated_at	timestampz

Рисунок 2.14 – Структура таблиці ігор (games) бази даних Gamestore

Сутність orders (замовлення)

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						46
Змн.	Арк.	№ докум.	Підпис	Дата		

orders		
id	uuid	
order_number	text	
user_id	uuid	
email	text	
status	text	
subtotal	numeric	
tax	numeric	
total	numeric	
created_at	timestampz	
payment_method	text	
liqpay_payment_id	text	
liqpay_status	text	
paid_at	timestampz	

Рисунок 2.15 – Структура таблиці замовлень (orders) бази даних Gamestore

Сутність order_items (позиції замовлення)

order_items		
id	uuid	
order_id	uuid	
game_id	text	
game_title	text	
quantity	int4	
unit_price	numeric	
line_total	numeric	
activation_key	text	

Рисунок 2.16 – Структура таблиці купленого товару бази даних Gamestore

Сутність refund_requests (заявки на повернення)

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

refund_requests		
id	uuid	
order_id	uuid	
user_id	uuid	
email	text	
reason	text	
status	text	
admin_comment	text	
created_at	timestampz	
updated_at	timestampz	

Рисунок 2.17 – Структура таблиці повернення товару бази даних Gamestore

Сутність password_reset_tokens (токени скидання пароля)

password_reset_tokens		
id	uuid	
user_id	uuid	
token	text	
expires_at	timestampz	
used	bool	
created_at	timestampz	

Рисунок 2.18 – Структура таблиці скидання паролю бази даних Gamestore

2.4 Висновки по розділу:

Спроековано архітектуру клієнт-серверної системи. Розроблено UML-діаграми та ER-модель бази даних. Обґрунтовано вибір технологічного стеку, який враховує сучасні вимоги до продуктивності та масштабованості.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ІНТЕРНЕТ-МАГАЗИНУ ПРОДАЖУ КОМП'ЮТЕРНИХ ІГОР

3.1. Налаштування та конфігурація проєкту

Правильне налаштування середовища розробки та конфігурація проєкту є критично важливим етапом, що забезпечує коректну роботу всіх компонентів системи як у локальному середовищі, так і на продакшен-сервері. У цьому підрозділі описано процес створення проєкту, встановлення залежностей, налаштування змінних середовища та конфігурацію основних інструментів.

Таблиця 3.1

Основні залежності

Next	15.x	Фреймворк
react / react-dom	19.x	UI-бібліотека
@supabase/supabase-js	2.x	БД-клієнт
@supabase/ssr	0.x	SSR-адаптер
@reduxjs/toolkit	2.x	Стан кошика

Таблиця 3.2

Змінні середовища

Змінна	Доступність	Обов'яз.	Опис:
NEXT_PUBLIC_SUPABASE_URL	Клієнт + сервер	Так	URL проєкту Supabase
NEXT_PUBLIC_SUPABASE_ANON_KEY	Клієнт + сервер	Так	Публічний anon-ключ
SUPABASE_SERVICE_ROLE_KEY	Лише сервер	Так	Сервісний ключ (обходить RLS)
NEXT_PUBLIC_LIQPAY_PUBLIC_KEY	Клієнт + сервер	так	Публічний ключ LiqPay

Усі чутливі параметри зберігаються у файлі `.env.local`, який не додається до системи контролю версій. Для спрощення розгортання в репозиторії міститься файл `.env.example` зі списком усіх необхідних змінних без значень, , що дозволяє новим розробникам швидко налаштувати локальне середовище.

3.1.1. Створення проєкту Next.js

Основою розробленого застосунку є фреймворк Next.js, ініціалізований з підтримкою TypeScript та Tailwind CSS. Структуру проєкту організовано за принципом App Router без додаткової папки src/, що забезпечує плоску та зрозумілу ієрархію маршрутів. Конфігурацію середовища розробки налаштовано відповідно до вимог системи: визначено змінні середовища для підключення до Supabase. Додаткові залежності встановлено згідно з архітектурними рішеннями, прийнятими на етапі проєктування: бібліотеки для керування станом (Redux Toolkit), автентифікації (NextAuth.js), валідації даних (Zod) та взаємодії з хмарною базою даних (Supabase JS).

Файл next.config.ts визначає основні параметри збірки та поведінку фреймворку:

Лістинг 3.1 - Налаштування конфігу

```
// next.config.ts

import type { NextConfig } from 'next';

const nextConfig: NextConfig = {

  images: {

    remotePatterns: [

      {

        protocol: 'https',

        hostname: '*.supabase.co', // зображення з Supabase Storage

        pathname: '/storage/v1/object/public/**',

      },

    ],

  },

  experimental: {
```

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

```
serverActions: { allowedOrigins: ['localhost:3000'] },
},
};
export default nextConfig;
```

3.2. Представлення роботи з базою даних

У цьому підрозділі розглядається організація роботи з базою даних Supabase (PostgreSQL), включаючи створення клієнтів для взаємодії з БД, виконання запитів, міграції схеми та обробку даних.

3.2.1. Клієнти Supabase

Для роботи з Supabase у проєкті створено два типи клієнтів:

Клієнт для браузера (lib/supabase/client.ts):

Використовує NEXT_PUBLIC_SUPABASE_ANON_KEY та має обмежені права. Використовується для клієнтських запитів, які не потребують спеціальних привілеїв [7].

Серверний клієнт (lib/supabase/server.ts):

Використовує SUPABASE_SERVICE_ROLE_KEY та має повний доступ до бази даних (обходить Row Level Security).

Лістинг 3.2 – Налаштування серверного компоненту

```
// lib/supabase/server.ts
import { createServerClient } from '@supabase/ssr';
import { cookies } from 'next/headers';

export function createClient() {
  return createServerClient(
```

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

```

process.env.NEXT_PUBLIC_SUPABASE_URL!,
process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,
{
  get(name: string) {
    return cookieStore.get(name)?.value;
  },
  set(name: string, value: string, options: any) {
    cookieStore.set({ name, value, ...options });
  },
  remove(name: string, options: any) {
    cookieStore.set({ name, value: "", ...options });
  }, }, }); }
export function createAdminClient() {
  return createServerClient(
    process.env.NEXT_PUBLIC_SUPABASE_URL!,
    process.env.SUPABASE_SERVICE_ROLE_KEY!,
    { cookies: {}, } ); }

```

3.3. Реалізація інтерфейсу системи продажу комп'ютерних ігор

У цьому підрозділі детально описується реалізація інтерфейсу користувача веб-застосунку "GameStore", включаючи адміністративну панель та клієнтську частину. Інтерфейс розроблено з використанням React, Next.js App Router, Tailwind CSS та бібліотеки компонентів.

3.3.1. Інтерфейс адміністратора

Адміністративна панель є закритою частиною системи, доступною лише користувачам з прапорцем `is_admin = true`. Вона надає інструменти для

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

керування каталогом ігор, користувачами, замовленнями, заявками на повернення та перегляду аналітики.

Структура адмін-панелі:

/admin

- └─ /login – сторінка входу для адміністратора
- └─ / – дашборд (KPI та діаграми)
- └─ /games – керування іграми (CRUD)
- └─ /users – керування користувачами (блокування)
- └─ /orders – керування замовленнями (зміна статусу)
- └─ /analytics – аналітика продажів (графіки за період)
- └─ /refunds – керування заявками на повернення

Дашборд (сторінка /admin):

Дашборд (сторінка /admin): Дашборд відображає ключові показники ефективності (KPI) у вигляді карток та діаграм. Кожна картка містить агреговані дані за останні 7 днів — загальну виручку, кількість замовлень, нових користувачів та найпопулярніші ігри — з відображенням динаміки відносно попереднього тижня у відсотках. Окрім карток, дашборд містить графіки продажів у розрізі часу та таблицю останніх замовлень, що дозволяє адміністратору оперативно відстежувати поточний стан платформи та виявляти тенденції. Реалізацію компонента наведено на рисунку 3.1. Динаміка у відсотках обчислюється шляхом порівняння поточного тижневого показника з аналогічним значенням попереднього тижня: позитивне значення позначається зеленим кольором, від'ємне — червоним, що дає змогу миттєво оцінити напрямок змін. Блок найпопулярніших ігор формується на основі агрегації даних таблиці order_items і відображає ігри з найбільшою кіл кількістю продажів за обраний період, забезпечуючи адміністратора актуальною інформацією про попит. Таким чином, дашборд слугує єдиною точкою моніторингу платформи, поєднуючи фінансову аналітику. Для забезпечення коректності та актуальності відображуваних даних усі запити до бази даних

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

виконуються на стороні сервера під час завантаження сторінки, що унеможливлює отримання застарілої інформації з кешу.

```

8  export default function AdminLoginPage() {
9    const [email, setEmail] = useState("");
10   const [password, setPassword] = useState("");
11   const [error, setError] = useState("");
12   const [loading, setLoading] = useState(false);
13
14   const handleSubmit = async (e: React.FormEvent) => {
15     e.preventDefault();
16     setError("");
17     setLoading(true);
18     try {
19       const res = await signIn("credentials", {
20         email,
21         password,
22         redirect: false,
23         callbackUrl: "/admin",
24       });
25       if (res?.error) {
26         setError("Невірний email або пароль, або акаунт заблоковано.");
27         setLoading(false);
28         return;
29       }
30       window.location.href = "/admin";
31     } catch {
32       setError("Помилка входу. Спробуйте ще раз.");
33       setLoading(false);
34     }
35   };

```

Рисунок 3.1 – Реалізація логіки клієнтського компонента завантаження даних адмін-панелі з використанням React Hooks (useEffect, useState)

Лістинг 3.3 - Компонент дашборду з візуалізацією

```

type Props = { data: AdminDashboardPayload };

function formatShortDate(iso: string) {

  return iso.slice(5); }

export default function AdminDashboardClient({ data }: Props) {

  const topForBar = data.popularGames.slice(0, 6).map((g) => ({

    name:

      g.title.length > 28 ? `${g.title.slice(0, 26)}...` : g.title,

    units: g.unitsSold, }));

  const deltaPct =

```

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

```

data.revenuePrevious7Days > 0

? ((data.revenueLast7Days - data.revenuePrevious7Days) /

    data.revenuePrevious7Days) *

    100

: data.revenueLast7Days > 0

? 100

: 0; }

```

Керування іграми (сторінка /admin/games):

Сторінка надає CRUD-функціональність для керування каталогом ігор: перегляд списку, додавання нової гри, редагування та м'яке видалення. При завантаженні сторінки виконується запит до бази даних, який повертає повний список ігор із відображенням ключових полів: назви, розробника, ціни, знижки та поточного статусу активності. Для додавання нової гри передбачено форму з валідацією всіх обов'язкових полів, що унеможливорює збереження неповних або некоректних даних. Редагування здійснюється через окрему форму, попередньо заповнену поточними значеннями запису, що дозволяє вносити точкові зміни без повторного введення всіх даних. М'яке видалення реалізовано через зміну значення поля `is_active` на `false`, завдяки чому гра зникає з публічного каталогу, але зберігається в базі даних для можливого відновлення в майбутньому. Реалізацію компонента показано на рисунку 3.2.

Окрім основних операцій, компонент забезпечує зручну навігацію між режимами перегляду та редагування завдяки динамічному рендерингу форм без перезавантаження сторінки. Усі зміни відображаються в таблиці в режимі реального часу після підтвердження операції, що забезпечує актуальність даних без необхідності ручного оновлення. Обробка помилок реалізована на рівні клієнтської валідації та серверних відповідей, що дозволяє своєчасно інформувати користувача про некоректні дії або збої під час взаємодії з базою даних. Додатково реалізовано

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

підсвічування рядків таблиці залежно від статусу активності гри, що дозволяє адміністратору миттєво візуально розрізняти активні та деактивовані записи.

```
app > admin > (shell) > games > pagetx > ...
25   export default function AdminGamesPage() {
149   return (
150     <div>
151       <h1 style={{ fontSize: "1.75rem", fontWeight: 700 }}>Управління іграми</h1>
152       <p className={styles.muted}>
153         Додавання, редагування та деактивація ігор у каталозі Supabase.
154       </p>
155       {message && <p style={{ marginTop: "0.75rem" }}>{message}</p>}
156       <h2 style={{ marginTop: "1.5rem" }}>
157         {editingId ? "Редагування" : "Нова гра"}
158       </h2>
159       <form className={styles.form} onSubmit={submit}>
160         <div>
161           <label>ID (slug, унікальний)</label>
162           <input
163             value={form.id}
164             onChange={(e) => setForm({ ...form, id: e.target.value })}
165             required
166             disabled={Boolean(editingId)}
167           />
168         </div>
169         <div>
170           <label>Назва</label>
171           <input
172             value={form.title}
173             onChange={(e) => setForm({ ...form, title: e.target.value })}
174             required
175           />
176         </div>
177       </form>
178     </div>
```

Рисунок 3.2 – Реалізація клієнтського компонента адміністрування ігор із формою додавання/редагування

Керування користувачами (сторінка /admin/users):

Сторінка дозволяє переглядати зареєстрованих користувачів, блокувати/розблоковувати їх та переглядати замовлення конкретного користувача. Реалізацію компонента наведено на рисунку 3.3

```
102   return (
103     <div>
104       <h1 style={{ fontSize: "1.75rem", fontWeight: 700 }}>
105         Управління користувачами
106       </h1>
107       <p className={styles.muted}>
108         Перегляд списку, блокування облікових записів та історія замовлень.
109       </p>
110       {message && <p style={{ marginTop: "0.75rem" }}>{message}</p>}
111       <div className={styles.tableWrap}>
112         <table className={styles.table}>
113           <thead>
114             <tr>
115               <th>Email</th>
116               <th>Ім'я</th>
117               <th>Адмін</th>
118               <th>Статус</th>
119             </tr>
120           </thead>
121           <tbody>
122             <tr>
123               <td>
124                 <div>
```

Рисунок 3.3 – Реалізація клієнтського компонента адміністрування користувачів із таблицею облікових записів

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3.2. Інтерфейс користувача (клієнта)

Клієнтська частина застосунку доступна всім відвідувачам (гостям) та зареєстрованим користувачам. Вона включає каталог ігор, сторінку товару, кошик, оформлення замовлення, бібліотеку покупок, історію замовлень та інші функції.

Структура клієнтської частини:

- | — / — головна сторінка (hero, мега-розпродажі, рекомендовані)
- | — /store — каталог ігор (фільтри, пошук, пагінація)
- | — /store/p/[game] — сторінка товару (опис, ціна, кнопка покупки)
- | — /cart — кошик (список товарів, підсумок)
- | — /checkout — оформлення замовлення (форма, оплата)
- | — /checkout/result — результат оплати
- | — /library — бібліотека придбаних ігор
- | — /orders — історія замовлень
- | — /refunds — заявки на повернення
- | — /account — обліковий запис (профіль, зміна пароля)
- | — /saved — збережені ігри (wishlist)
- | — /about — про нас
- | — /support — підтримка (зворотний зв'язок)
- | — /terms, /privacy — правові сторінки

Головна сторінка (app/page.tsx):

Головна сторінка містить hero-секцію, карусель мега-розпродажів, блоки рекомендованих ігор та новинок.

Лістинг 3.4 -Головна Сторінка

```
import SaleGamesCarousel from "@components/ui/carusel/SaleGamesCarousel";  
  
import BestOffer from "@components/ui/grid/BestOffer";  
  
import { MegaSaleSection } from "@components/MegaSale/MegaSaleSection";
```

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import { loadCatalogGames } from "@lib/catalog/loadGames";

import { HomeContent } from "@components/home/HomeContent";

export const revalidate = 300;

export default async function Home() {

  const saleGames = await loadCatalogGames();

  return (

    <HomeContent>

      <SaleGamesCarousel games={saleGames} />

      <BestOffer />

      <MegaSaleSection games={saleGames} />

    </HomeContent>

  ); }

```

Каталог ігор (app/store/page.tsx):

Каталог реалізований як серверний компонент з підтримкою фільтрації та пагінації через URL-параметри.

Лістинг 3.5 – сторінка каталогу

```

Import { loadCatalogGames } from "@lib/catalog/loadGames";

import StorePageClient from "../StorePageClient";

export const revalidate = 300;

export default async function StorePage() {

  const games = await loadCatalogGames();

```

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

```
return <StorePageClient games={games} />;

}
```

Сторінка товару (app/store/p/[game]/page.tsx):

Сторінка товару відображає детальну інформацію про гру, включаючи назву, опис, ціну, обкладинку, галерею скріншотів, жанр, платформу, рік випуску, теги та кнопку додавання до кошика. Сторінка реалізована як серверний компонент Next.js, що дозволяє виконувати запит до бази даних безпосередньо на сервері та передавати готові дані у клієнтські компоненти. Отримання даних виконується через параметр маршруту `params.game`, який містить `slug` гри. Запит до Supabase здійснюється на рівні серверного компонента, а у разі відсутності запису викликається `notFound()` для відображення сторінки 404. Такий підхід забезпечує оптимальну продуктивність сторінки, оскільки весь процес отримання та обробки даних відбувається на сервері ще до відправки відповіді клієнту, що значно скорочує час першого відображення контенту. Лістинг коду наведено на рисунку 3.7

```
export default function GamePage() {
  const params = useParams();
  const gameId = params.game as string;
  const dispatch = useDispatch();
  const router = useRouter();
  const [game, setGame] = useState<Game | null | undefined>(undefined);

  const recentGameData = useMemo(
    () => {
      game
      ? {
        id: game.id,
        title: game.title,
        imageUrl: game.imageUrl,
        originalPrice: game.originalPrice,
        discountedPrice: game.discountedPrice,
        discount: game.discount,
      }
      : null,
    [game?.id, game?.title, game?.imageUrl, game?.originalPrice, game?.discountedPrice, game?.discount],
  );

  const recentlyViewed = useRecentlyViewed(gameId, recentGameData);
```

Рисунок 3.7 – Лістинг коду динамічної сторінки товару (page.tsx) з отриманням даних за параметром маршруту

Кошик (app/cart/page.tsx):

Кошик використовує Redux для зберігання стану з персистенцією (redux-persist). Лістинг коду наведено на рисунку 3.8

```
16 export default function CartPage() {
17   const { data: session, status } = useSession();
18   const items = useSelector((state: RootState) => state.cart.items);
19   const dispatch = useDispatch();
20   const isAuthenticated = status === "authenticated" && !!session;
21
22   // Коректний парсер ціни
23   const parsePrice = (price: string) => {
24     const digitsOnly = price.replace(/[^d]/g, "");
25     return Number(digitsOnly) / 100;
26   };
27
28   // Коректна загальна сума
29   const totalPrice = items.reduce((sum, item) => {
30     return sum + parsePrice(item.discountedPrice) * item.quantity;
31   }, 0);
32
33   // Порожній кошик
34   if (items.length === 0) {
35     return (
36       <>
37       <h1 className={styles.title}>Мій Кошик</h1>
38       <main className={styles.container}>
39         <div className={styles.emptyState}>
40           <h1 className={styles.emptyText}>Ваш кошик поки що порожній.</h1>
41           <Link href="/" className={styles.button}>
42             Перейти до магазину
43           </Link>
44         </div>
45       </main>
46     </>
47   )
48 }
```

Рисунок 3.8 – Лістинг коду сторінки кошика (app/cart/page.tsx) з управлінням станом через Redux

Лістинг коду наведено на рисунку 3.9

```
export const cartSlice = createSlice({
  name: "cart",
  initialState,
  reducers: {
    addToCart: (state, action: PayloadAction<Omit<CartItem, "quantity">>) => {
      const existingItem = state.items.find(
        (item) => item.id === action.payload.id,
      );
      if (existingItem) { ... }
      else {
        state.items.push(...);
      }
    },
    increaseQuantity: (state, action: PayloadAction<string>) => { ... },
    decreaseQuantity: (state, action: PayloadAction<string>) => { ... },
    removeFromCart: (state, action: PayloadAction<string>) => { ... },
    clearCart: (state) => {
      state.items = [];
    },
  },
});
```

Рисунок 3.9 – Лістинг Redux slice для управління станом кошика (lib/store/cartSlice.ts)

					БР.ІП - 07.00.00.000 ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

Оформлення замовлення (app/checkout/page.tsx):

Сторінка оформлення замовлення включає форму контактних даних, поле для промокоду та інтеграцію з LiqPay. Лістинг коду наведено на рисунку 3.10.

```
export default function CheckoutPage() {
  const form = useCheckoutForm();

  if (form.sessionStatus === "loading") {
    return (
      <div className={styles.container}>
        <div className={cStyles.emptyState}>
          <div className={cStyles.loaderRing} />
          <p className={cStyles.emptyText}>Завантаження...</p>
        </div>
      </div>
    );
  }

  if (form.sessionStatus === "unauthenticated" || !form.session) {
    return (
      <div className={styles.container}>...
    </div>
    );
  }

  if (form.items.length === 0 && !form.showReceipt) {
    return (...);
  }
}
```

Рисунок 3.10 – Лістинг коду сторінки оформлення замовлення

Бібліотека покупок (app/library/page.tsx):

Сторінка бібліотеки відображає всі ігри, придбані користувачем.

3.4 Висновки по розділу

Реалізовано повнофункціональний веб-застосунок з каталогом, кошиком, платіжним шлюзом та адміністративною панеллю, які в сукупності забезпечують повний цикл взаємодії користувача з платформою. Клієнтська частина побудована на основі Next.js з використанням App Router, що забезпечує високу продуктивність завдяки серверному рендерингу та оптимізованому завантаженню сторінок. Серверна логіка реалізована через API-маршрути та інтеграцію з Supabase, що дозволяє обробляти запити.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ІНТЕРНЕТ-МАГАЗИНУ ПРОДАЖУ КОМ'ЮТЕРНИХ ІГОР

4.1. Налаштування та конфігурація проєкту

Правильне налаштування середовища розробки та конфігурація проєкту є критично важливим етапом, що забезпечує коректну роботу всіх компонентів системи як у локальному середовищі, так і на продакшен-сервері. У цьому підрозділі описано процес створення проєкту, встановлення залежностей, налаштування змінних середовища та конфігурацію основних інструментів [5].

Основні залежності проєкту включають: `next (15.x)` — фреймворк; `react / react-dom (19.x)` — UI-бібліотека; `@supabase/supabase-js (2.x)` — клієнт бази даних; `@supabase/ssr (0.x)` — SSR-адаптер; `@reduxjs/toolkit (2.x)` — керування станом кошика [9].

Усі чутливі параметри зберігаються у файлі `.env.local`, який не додається до системи контролю версій [8]. Змінні середовища включають: `NEXT_PUBLIC_SUPABASE_URL` та `NEXT_PUBLIC_SUPABASE_ANON_KEY` — для підключення до Supabase (доступні на клієнті та сервері); `SUPABASE_SERVICE_ROLE_KEY` — сервісний ключ, доступний лише на сервері, обходить Row Level Security; `NEXT_PUBLIC_LIQPAY_PUBLIC_KEY` — публічний ключ платіжного шлюзу. Для спрощення розгортання в репозиторії міститься файл `.env.example` зі списком усіх необхідних змінних без значень [24].

4.1.1. Створення проєкту Next.js

Основою розробленого застосунку є фреймворк Next.js, ініціалізований з підтримкою TypeScript та Tailwind CSS. Структуру проєкту організовано за принципом App Router без додаткової папки `src/`, що забезпечує плоску та зрозумілу ієрархію маршрутів [5]. Конфігурацію середовища розробки налаштовано відповідно до вимог системи: визначено змінні середовища для

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		62

підключення до Supabase, платіжного шлюзу LiqPay та сервісів автентифікації [15]. Додаткові залежності встановлено згідно з архітектурними рішеннями, прийнятими на етапі проєктування: бібліотеки для керування станом (Redux Toolkit), автентифікації (NextAuth.js), валідації даних (Zod) та взаємодії з хмарною базою даних (Supabase JS) [9].

Файл `next.config.ts` визначає основні параметри збірки та поведінку фреймворку, зокрема дозволяє завантаження зображень зі сховища Supabase Storage та налаштовує Server Actions для локального середовища розробки [24].

4.2. Представлення роботи з базою даних

У цьому підрозділі розглядається організація роботи з базою даних Supabase (PostgreSQL), включаючи ініціалізацію клієнта, виконання запитів, фільтрацію даних та обробку помилок. Supabase надає зручний JavaScript-клієнт, який абстрагує пряму роботу з SQL і дозволяє виконувати операції з таблицями через ланцюжок методів.

Ініціалізація клієнта виконується один раз у окремому модулі та імпортується у сервісні шари застосунку. Параметри підключення зберігаються у змінних середовища, що забезпечує безпеку облікових даних.

4.2.1. Клієнти Supabase

Для роботи з Supabase у проєкті створено два типи клієнтів. Клієнт для браузера (`lib/supabase/client.ts`) використовує `NEXT_PUBLIC_SUPABASE_ANON_KEY` та має обмежені права — застосовується для клієнтських запитів, які не потребують спеціальних привілеїв [7]. Серверний клієнт (`lib/supabase/server.ts`) використовує `SUPABASE_SERVICE_ROLE_KEY` та має повний доступ до бази даних, обходячи Row Level Security — використовується в API-маршрутах та адміністративних скриптах. Додатково реалізовано функцію `createAdminClient()` для операцій, що потребують підвищених прав доступу незалежно від контексту запиту.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

4.3. Реалізація інтерфейсу системи продажу комп'ютерних ігор

У цьому підрозділі детально описується реалізація інтерфейсу користувача веб-застосунку "GameStore", включаючи адміністративну панель та клієнтську частину. Інтерфейс розроблено з використанням React, Next.js App Router, Tailwind CSS та бібліотеки компонентів.

4.3.1. Інтерфейс адміністратора

Адміністративна панель є закритою частиною системи, доступною лише користувачам з прапорцем `is_admin = true`. Доступ до маршрутів `/admin/*` захищено middleware [9]. Панель надає інструменти для керування каталогом ігор, користувачами, замовленнями, заявками на повернення та перегляду аналітики.

Структура адмін-панелі включає такі маршрути: `/admin/login` — сторінка входу для адміністратора; `/admin` — дашборд з KPI та діаграмами; `/admin/games` — керування іграми (CRUD); `/admin/users` — керування користувачами (блокування); `/admin/orders` — керування замовленнями (зміна статусу); `/admin/analytics` — аналітика продажів (графіки за період); `/admin/refunds` — керування заявками на повернення коштів.

4.3.2. Інтерфейс користувача (клієнта)

Клієнтська частина застосунку доступна всім відвідувачам (гостям) та зареєстрованим користувачам. Вона включає каталог ігор, сторінку товару, кошик, оформлення замовлення, бібліотеку покупок, історію замовлень та інші функції.

Структура клієнтської частини містить такі маршрути: `/` — головна сторінка; `/store` — каталог ігор з фільтрацією, пошуком та пагінацією; `/store/p/[game]` — сторінка товару (опис, ціна, зображення, кнопка покупки); `/cart` — кошик; `/checkout` — оформлення замовлення; `/checkout/result` — результат оплати; `/library` — бібліотека придбаних ігор; `/orders` — історія замовлень; `/refunds` — заявки на повернення коштів; `/account` — обліковий запис (профіль, зміна

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

пароля); /saved — збережені ігри (wishlist); /support — підтримка (зворотний зв'язок) [5].

4.1.3 Реалізація каталогу ігор

Каталог ігор забезпечує перегляд продукції, фільтрацію за жанром і платформою, сортування та посторінкову навігацію. Стан фільтрів та поточна сторінка зберігаються у параметрах URL через хук useSearchParams.

Лістинг 4.1 -Сторінка фільтрів

```
const [searchParams, setSearchParams] = useSearchParams();

const [games, setGames] = useState([]);

useEffect(() => {

  const filters = {

    genre: searchParams.get('genre'),

    platform: searchParams.get('platform'),

    sort: searchParams.get('sort') || 'newest',

    page: searchParams.get('page') || 1,

    est',

    page: searchParams.get('page') || 1,

  };

  GameService.getAll(filters).then(setGames);

}, [searchParams]);
```

Ігри відображаються у вигляді карток із обкладинкою, назвою, жанром та ціною.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```
function GameCard({ game }) {
  return (
    <div className="game-card">
      <img src={game.coverUrl} alt={game.title} />
      <h3>{game.title}</h3>
      <span className="genre">{game.genre}</span>
      <p className="price">{game.price} грн</p>
      <AddToCartButton gameId={game.id} />
    </div>
  );
}
```

4.1.4 Реалізація сторінки гри

Сторінка гри використовується для детального перегляду інформації про продукт перед оформленням покупки. На сторінці відображаються обкладинка та галерея скріншотів, повний опис гри, жанр, платформа, рік випуску, системні вимоги, а також відгуки користувачів з оцінками.

Отримання даних виконується через хук `useParams` для отримання ідентифікатора гри з URL-адреси та хук `useEffect` для надсилання запиту до API при першому завантаженні компонента. Під час очікування відповіді відображається індикатор завантаження, а у разі помилки — повідомлення про неможливість отримати дані. Після успішного отримання інформації стан компонента оновлюється та сторінка перерендерюється з актуальними даними.

Лістинг 4.2 – Хук даних

```
const { id } = useParams();
```

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

```
const [game, setGame] = useState(null);
```

```
useEffect(() => {  
  GameService.getById(id).then(setGame);  
}, [id]);
```

Для перегляду скріншотів реалізовано галерею із зміною активного зображення при кліку на мініатюру.

```
const [activeImg, setActiveImg] = useState(0);
```

```
<div className="gallery">  
  <img src={game.screenshots[activeImg]} className="main-img"  
alt={`Screenshot ${activeImg + 1}`} />  
  <div className="thumbnails">  
    {game.screenshots.map((src, i) => (  
      <img key={i} src={src}  
        className={i === activeImg ? 'active' : ''}  
        onClick={() => setActiveImg(i)}  
        alt={`Thumbnail ${i + 1}`} />  
    ))}  
  </div>  
  <div className="gallery-counter">  
    {activeImg + 1} / {game.screenshots.length}  
  </div>  
</div>
```

Реалізацію галереї показано на рисунку 4.6

					БР.ІІІ - 07.00.00.000 ІІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

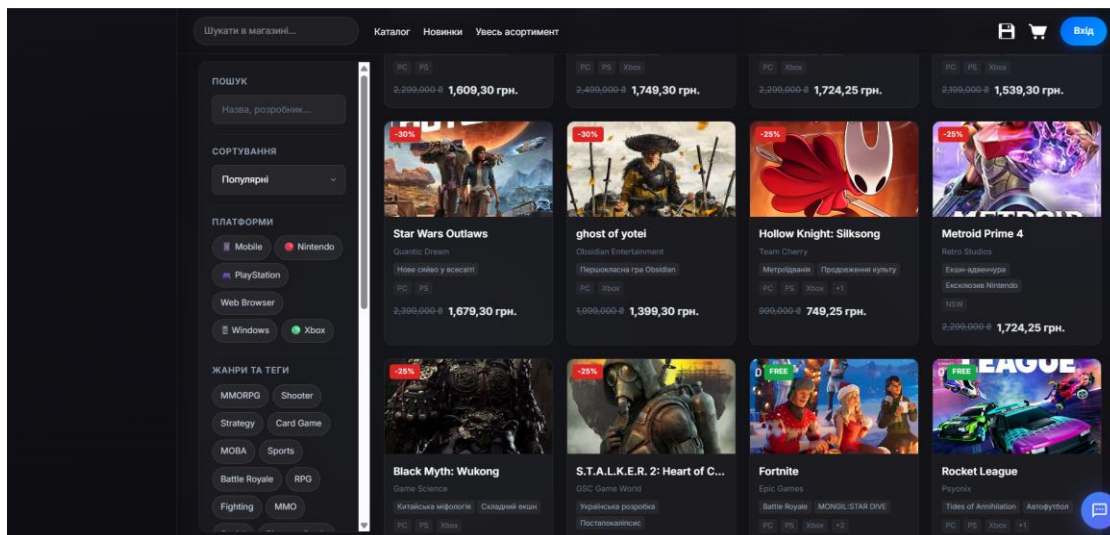


Рисунок 4.6 – Галерея скріншотів гри з мініатюрами

Основний блок містить назву, рейтинг, ціну та кнопку додавання до кошика.

Лістинг 4.4 -Об'єкт даних

```
<div className="game-info">
  <h1 className="game-title">{game.title}</h1>

  <div className="game-meta">
    <span className="game-genre">{game.genre}</span>
    <span className="game-platform">{game.platform}</span>
    <span className="game-year">{game.releaseYear}</span>
  </div>

  <div className="game-rating">
    <span className="stars">
      {[1, 2, 3, 4, 5].map(star => (
        <span key={star} className={star <= Math.round(game.rating) ? 'star filled'
: 'star'}>
      </span>
      <span className="rating-value">{game.rating} / 5</span>
    )}
  </div>
</div>
```

					БР.ІІІ - 07.00.00.000 ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		68

```

<span className="reviews-count">({game.reviewsCount} відгуків)</span>
</div>

```

Інформаційний блок сторінки показано на рисунку 4.7.

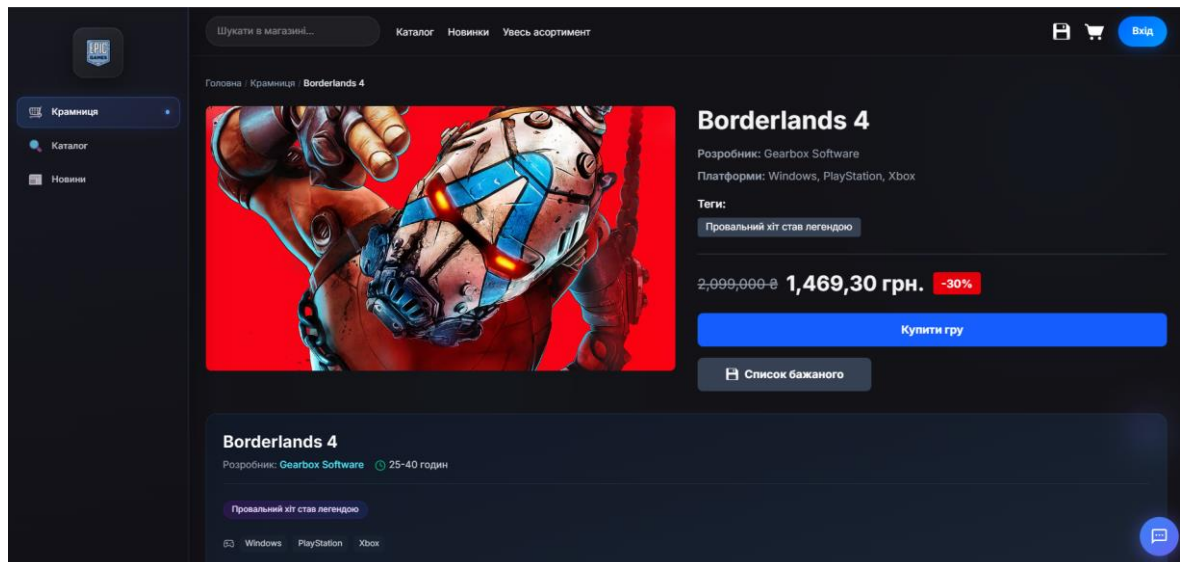


Рисунок 4.7 – Інформаційний блок сторінки з кнопкою додавання до кошика

4.4. Висновки по розділу

У даному розділі реалізовано повнофункціональний веб-застосунок інтернет-магазину з продажу комп'ютерних ігор "GameStore". Налаштовано середовище розробки з використанням Next.js 14, TypeScript, Supabase та Tailwind CSS [8]. Реалізовано два типи клієнтів Supabase — браузерний та серверний — для безпечної роботи з базою даних відповідно до рівня привілеїв..Реалізовано клієнтську частину застосунку. Забезпечено високий рівень безпеки системи через зберігання секретів у змінних середовища [11].

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		69

РОЗДІЛ 5. ТЕСТУВАННЯ ВЕБ-ЗАСТОСУНКУ

Тестування є важливим етапом розробки веб-застосунку, оскільки дозволяє перевірити коректність роботи функціоналу, стабільність системи та відповідність реалізованих можливостей встановленим вимогам [4]. Під час тестування перевіряється робота користувацької та адміністративної частин застосунку, а також коректність виконання основних операцій.

Для перевірки веб-застосунку використовуються автоматизовані та функціональні методи тестування, які допомагають виявити можливі помилки та оцінити стабільність роботи системи [16].

5.1 Тестування користувацької частини веб-застосунку

Тестування веб-застосунку дає можливість оцінити правильність роботи основного функціоналу та взаємодії користувача із системою. Основна увага приділяється користувацькому інтерфейсу, зокрема навігації, роботі з каталогом товарів, перегляду сторінок і взаємодії з кошиком [6]. Також перевіряється правильність виконання основних операцій користувача та стабільність роботи інтерфейсу [4].

5.1.1 Автоматизоване тестування інтерфейсу

Для перевірки користувацького інтерфейсу використовується автоматизоване E2E тестування за допомогою інструменту Puppeteer. Воно запускається в середовищі розгорнутого застосунку, що дозволяє перевірити роботу системи в умовах, максимально наближених до реального використання. Такий підхід дає змогу зрозуміти, чи інтерфейс працює коректно і чи відповідає очікуванням користувачів. Автоматизовані тести імітують дії користувача в браузері та охоплюють основні сценарії роботи із системою.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

Зокрема, перевіряється:

- відкриття головної сторінки;
- перехід між сторінками застосунку;
- перегляд каталогу товарів;
- перехід на сторінку товару;
- наявність товарів у каталозі;
- робота з кошиком.

Тестові сценарії подані як набір перевірок, згрупованих за окремими функціональними частинами.

Фрагмент реалізації автоматизованого тестування наведено нижче.

Лістинг 5.1 - Фрагмент реалізації

```
describe('Робота з товарами', () => {  
  test('Каталог містить товари', async () => {  
    await page.goto(`${BASE_URL}/products`, {  
    });  
    const links = await page.$$('a[href*="/products/"]');  
    expect(links.length).toBeGreaterThan(0);  
  });  
  test('Відкриття сторінки товару', async () => {  
    await page.goto(`${BASE_URL}/products`);  
    const links = await page.$$('a[href*="/products/"]');  
    if (links.length > 0) {  
      await links[0].click();  
      await page.waitForNavigation();  
      expect(page.url()).toMatch(/products\/\d+/);  
    } else {  
      throw new Error('Немає товарів для тесту');  
    }  
  });  
});
```

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

Під час тестування автоматично відкриваються сторінки, виконуються дії користувача і перевіряються результати, що дозволяє оцінити коректність інтерфейсу та роботи системи [16].

5.1.2 Перевірка роботи каталогу та кошика

Під час тестування користувацької частини веб-застосунку також перевірялась робота каталогу товарів і кошика [6]. Основна увага приділялась правильному відображенню товарів, роботі фільтрації, сортування та зміні кількості товарів у кошику.

Додатково перевірялась правильність обчислення загальної суми замовлення та збереження стану кошика між переходами між сторінками [11]. У процесі тестування критичних помилок у роботі цих компонентів виявлено не було.

5.2 Тестування адміністративної частини

Тестування адміністративної частини проводиться для перевірки правильності роботи функцій управління даними та контролю доступу до системи [12]. Основна увага приділяється роботі з товарами, користувачами, замовленнями та повідомленнями, а також дотриманню бізнес-логіки під час виконання адміністративних операцій.

5.2.1 Функціональне тестування адміністративної частини

Для оцінки адміністративної частини веб-застосунку було проведено функціональне тестування, яке включає перевірку основних функцій системи без застосування автоматизованих інструментів [4].

У рамках тестування здійснено перевірку:

- авторизації адміністратора та доступу до адміністративної панелі;
- управління товарами (створення, редагування, видалення);
- управління користувачами (перегляд та зміна ролей);
- обробки замовлень та зміни їх статусів;

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

- роботи з повідомленнями користувачів;
 - дотримання обмежень бізнес-логіки під час зміни статусів замовлення.
- Результати тестування представлені у таблиці 5.1.

Таблиця 5.1

Результати функціонального тестування адміністративної панелі

№	Модуль	Сценарій	Дія	Очікуваний результат	Фактичний результат
1	Авторизація	Вхід в адмін-панель	Введення коректних даних	Успішний вхід в систему	Вхід виконано
2	Товари	Додавання товару	Заповнення форми та збереження	Створення нового товару	Виконано
3	Товари	Редагування товару	Зміна характеристик	Оновлення даних	Виконано
4	Товари	Видалення товару	Підтвердження видалення	Товар видалено	Виконано
5	Користувачі	Перегляд списку користувачів	Відкриття сторінки зі списком	Список відображається	Виконано
6	Користувачі	Редагування ролі користувача	Зміна ролі користувача	Вибір нової ролі та збереження	Виконано
7	Замовлення	Зміна статусу замовлення	Вибір допустимого статусу	Статус змінено	Виконано
8	Замовлення	Некоректна зміна статусу	Спроба обрати заборонений статус	Відображення помилки	Виконано
9	Повідомлення	Перегляд списку повідомлень користувачів	Відкриття звернення	Відображення даних	Виконано
10	Повідомлення	Відповідь на повідомлення користувача	Надсилання відповіді	Відповідь надіслано	Виконано

Проведене функціональне тестування показало правильну роботу основних модулів адміністративної панелі та підтвердило коректність виконання основних операцій. Отримані результати свідчать про стабільну роботу функцій управління даними та взаємодії між окремими компонентами системи.

5.2.2 Перевірка бізнес-логіки та обмежень доступу

Під час тестування окремо перевірялась правильність реалізації бізнес-логіки системи та механізмів контролю доступу [12]. Особлива увага приділялась зміні статусів замовлень, перевірці допустимих переходів між станами та обмеженню доступу до адміністративної частини застосунку [11].

Результати перевірки показали, що система правильно обробляє недопустимі дії користувачів і дотримується встановлених правил роботи бізнес-логіки.

5.3 Результати тестування та оцінка роботи системи

Після виконання тестових сценаріїв було проведено аналіз отриманих результатів та оцінку стабільності роботи веб-застосунку [4]. Основна увага приділялась правильному виконанню операцій, роботі інтерфейсу та взаємодії між окремими компонентами системи.

5.3.1 Аналіз результатів тестування

Під час тестування було встановлено, що веб-застосунок загалом правильно реалізує основні сценарії взаємодії користувача із системою. Автоматизовані перевірки показали, що інтерфейс працює стабільно, навігація не викликає проблем, а основні операції виконуються як очікується [16]. Також було перевірено правильність переходів між сторінками та відображення даних у різних частинах застосунку.

Критичних помилок під час виконання тестових сценаріїв виявлено не було. Результати тестування наведені на рисунку 5.1.

					БР.ІІІ - 07.00.00.000 ІІЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

PS C:\Users\Honor\Desktop\tests> npx jest
console.log
Кнопки не знайдені, тест пропущено

    at Object.log (e2e.test.js:108:17)

PASS ./e2e.test.js (21.258 s)
  E2E тестування користувацького інтерфейсу Celoria Shop
    Завантаження сторінок
      ✓ Головна сторінка відкривається (3049 ms)
      ✓ Каталог відкривається (2337 ms)
      ✓ Кошик відкривається (609 ms)
    Навігація користувача
      ✓ Перехід з головної у каталог (3424 ms)
    Робота з товарами
      ✓ Каталог містить товари (2926 ms)
      ✓ Відкриття сторінки товару (3569 ms)
    Робота з кошиком
      ✓ Додавання товару у кошик (2308 ms)
      ✓ Перехід до кошика після дій (363 ms)

Test Suites: 1 passed, 1 total
Tests:       8 passed, 8 total
Snapshots:   0 total
Time:        21.372 s, estimated 24 s
Ran all test suites.

```

Рисунок 5.1 – Результати автоматизованого тестування

Функціональне тестування адміністративної частини також показало коректну роботу механізмів управління даними та дотримання бізнес-логіки [11]. Отримані результати свідчать, що система працює стабільно і реалізований функціонал відповідає встановленим вимогам.

5.4 Висновки по розділу

Проведене тестування показало, що веб-застосунок правильно реалізує основні функціональні можливості та забезпечує стабільну взаємодію користувача із системою. Автоматизоване та функціональне тестування підтвердили правильну роботу користувацької й адміністративної частин застосунку.

Отримані результати свідчать про стабільну роботу системи та відповідність реалізованого функціоналу встановленим вимогам.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВОК

У процесі виконання дипломної роботи були визначені основні принципи функціонування систем електронної комерції та сформульовані вимоги до веб-застосунку інтернет-магазину. Аналіз сучасних рішень дозволив врахувати їхні переваги та обмеження при проектуванні системи.

Обґрунтовано вибір архітектури та технологій розробки, які забезпечують ефективність, структурованість і масштабованість застосунку. Використання архітектури MVC дозволяє розділити обробку даних, бізнес-логіку та інтерфейс користувача, що спрощує супровід і подальший розвиток системи.

Реалізовано веб-застосунок, що забезпечує повний цикл взаємодії користувача із системою — від перегляду товарів до оформлення замовлення. Інтерфейс організовано з акцентом на зручність використання, логічну навігацію та швидкий доступ до основного функціоналу.

Адміністративна частина забезпечує ефективне управління товарами, користувачами, замовленнями та повідомленнями. Реалізовані механізми контролю доступу та перевірки дій користувачів сприяють безпечній роботі системи та запобігають некоректним операціям.

Бізнес-логіка системи гарантує коректну обробку даних, узгодженість роботи компонентів і контроль виконання операцій. Реалізовані перевірки та обмеження дозволяють уникнути помилок під час роботи з даними та забезпечують стабільність функціонування системи.

Аналітичні модулі дозволяють отримувати, обробляти та візуалізувати ключові показники діяльності системи, що спрощує аналіз ефективності її роботи та підтримує прийняття обґрунтованих рішень щодо подальшого розвитку.

Результати тестування підтверджують працездатність, стабільність та відповідність застосунку встановленим вимогам. Отримані результати можуть бути використані для подальшого розвитку та практичного застосування системи.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Глушков В. М. Інформаційні технології та системи в економіці. — Київ : КНЕУ, 2021. — 312 с.
2. Ковальчук О. І. Електронна комерція: стан і перспективи розвитку в Україні // Економічний вісник. — № 3 (2023). — С. 45–53.
3. Sommerville I. Software Engineering. — 11th ed. — Pearson Education, 2020. — 792 p.
4. Марченко І. П. Цифрова дистрибуція відеоігор: сучасний стан та тенденції розвитку // Маркетинг і менеджмент інновацій. — № 2 (2022). — С. 85–94.
5. Next.js Documentation. App Router Overview. — [Електронний ресурс]. — Режим доступу: <https://nextjs.org/docs/app> — Дата звернення: 05.03.2026.
6. Chevalier J., Mayzlin D. The Effect of Word of Mouth on Sales: Online Book Reviews // Journal of Marketing Research. — 2006. — Vol. 43, No. 3. — P. 345–354.
7. Supabase Documentation. PostgreSQL Database Platform. — [Електронний ресурс]. — Режим доступу: <https://supabase.com/docs> — Дата звернення: 12.04.2026.
8. NextAuth.js Documentation. Authentication for Next.js. — [Електронний ресурс]. — Режим доступу: <https://next-auth.js.org> — Дата звернення: 20.05.2026.
9. OWASP Foundation. OWASP Cheat Sheet Series (Web Security). — [Електронний ресурс]. — Режим доступу: <https://cheatsheetseries.owasp.org/> — Дата звернення: 20.04.2026.
10. TypeScript Documentation. TypeScript Handbook. — [Електронний ресурс]. — Режим доступу: <https://www.typescriptlang.org/docs/> — Дата звернення: 20.05.2026.
11. Vercel. Next.js Deployment Guide. — [Електронний ресурс]. — Режим доступу: <https://vercel.com/docs/frameworks/nextjs> — Дата звернення: 10.03.2026.
12. Fowler M. Patterns of Enterprise Application Architecture. — Addison-Wesley, 2002. — 533 p.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Redux Toolkit Documentation. State Management for React. — [Електронний ресурс]. — Режим доступу: <https://redux-toolkit.js.org> — Дата звернення: 05.05.2026.

14. W3C. Web Design and Applications Standards. — [Електронний ресурс]. — Режим доступу: <https://www.w3.org/standards/webdesign/> — Дата звернення: 05.03.2026.

15. LiqPay API Documentation. Payment Gateway Integration. — [Електронний ресурс]. — Режим доступу: <https://www.liqpay.ua/documentation> — Дата звернення: 11.02.2026.

16. PostgreSQL Documentation. PostgreSQL 16 Database System. — [Електронний ресурс]. — Режим доступу: <https://www.postgresql.org/docs/16/> — Дата звернення: 20.04.2026.

17. Resend Documentation. Email API for Developers. — [Електронний ресурс]. — Режим доступу: <https://resend.com/docs> — Дата звернення: 20.03.2026.

18. Tailwind CSS Documentation. Utility-First CSS Framework. — [Електронний ресурс]. — Режим доступу: <https://tailwindcss.com/docs> — Дата звернення: 01.03.2026.

19. Zod Documentation. TypeScript-first Schema Validation. — [Електронний ресурс]. — Режим доступу: <https://zod.dev> — Дата звернення: 30.05.2026.

20. Recharts Documentation. Charts for React Applications. — [Електронний ресурс]. — Режим доступу: <https://recharts.org> — Дата звернення: 12.03.2026.

21. Richardson C. Microservices Patterns: With Examples in Java. — Manning Publications, 2018. — 520 p.

22. AndriyMykhailiuk. Репозиторій дипломного проєкту «Розробка масштабованого Full-Stack застосунку для продажу комп'ютерних ігор» (GameStore). — [Електронний ресурс]. — Режим доступу: https://github.com/AndriyMykhailiyk/EpicGameShop_V1 — Дата звернення: 10.06.2026.

23. Pressman R. S. Software Engineering: A Practitioner's Approach. — 9th ed. — McGraw-Hill, 2019. — 976 p.

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

24. Lauren S. Managing Digital Game Distribution: Platforms, Business Models, and User Behavior // Journal of Media Economics. — 2021. — Vol. 34, No. 1. — P. 22–39.

25. React Documentation. The React Library for User Interfaces. — [Електронний ресурс]. — Режим доступу: <https://react.dev> — Дата звернення: 01.06.2026.

26. Node.js Documentation. Node.js JavaScript Runtime. — [Електронний ресурс]. — Режим доступу: <https://nodejs.org/en/docs> — Дата звернення: 01.06.2026.

27. Stripe API Documentation. Global Payment Processing. — [Електронний ресурс]. — Режим доступу: <https://docs.stripe.com> — Дата звернення: 15.05.2026.

28. TanStack Query Documentation. Async State Management. — [Електронний ресурс]. — Режим доступу: <https://tanstack.com/query> — Дата звернення: 25.05.2026.

29. Jest Documentation. JavaScript Testing Framework. — [Електронний ресурс]. — Режим доступу: <https://jestjs.io/docs> — Дата звернення: 05.06.2026.

30. Newman S. Building Microservices: Designing Fine-Grained Systems. — 2nd ed. — O'Reilly Media, 2021. — 650 p.

31. Пасічник В. В., Пасічник О. В. Інформаційні системи та технології в електронній комерції. — Львів : Магнолія, 2020. — 280 с.

32. GitHub Documentation. GitHub Actions for CI/CD. — [Електронний ресурс]. — Режим доступу: <https://docs.github.com/en/actions> — Дата звернення: 10.06.2026.

33. Docker Documentation. Containerization Platform. — [Електронний ресурс]. — Режим доступу: <https://docs.docker.com> — Дата звернення: 12.05.2026.

34. Lighthouse Metrics (Google). Performance Best Practices for Modern Web Applications. — [Електронний ресурс]. — Режим доступу: <https://developer.chrome.com/docs/lighthouse/> — Дата звернення: 18.05.2026.

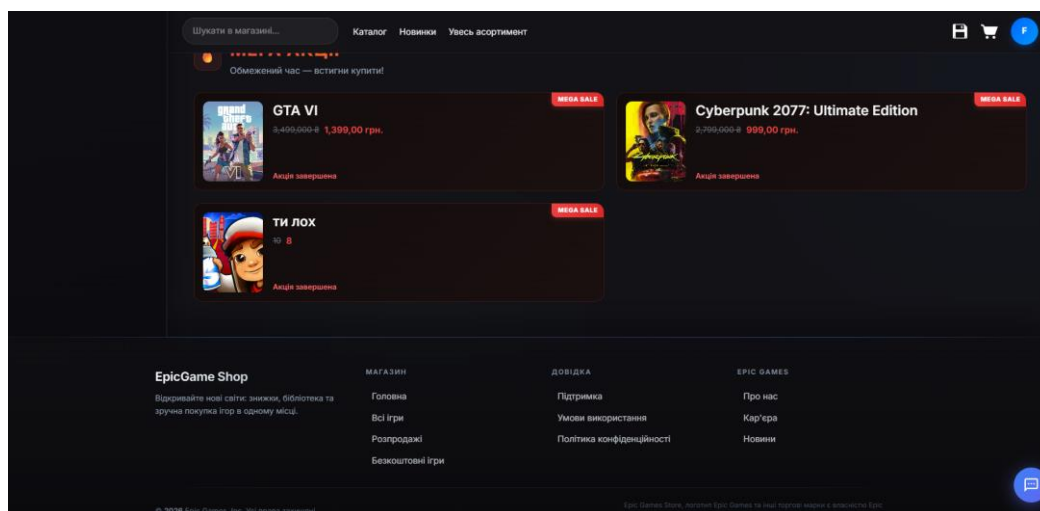
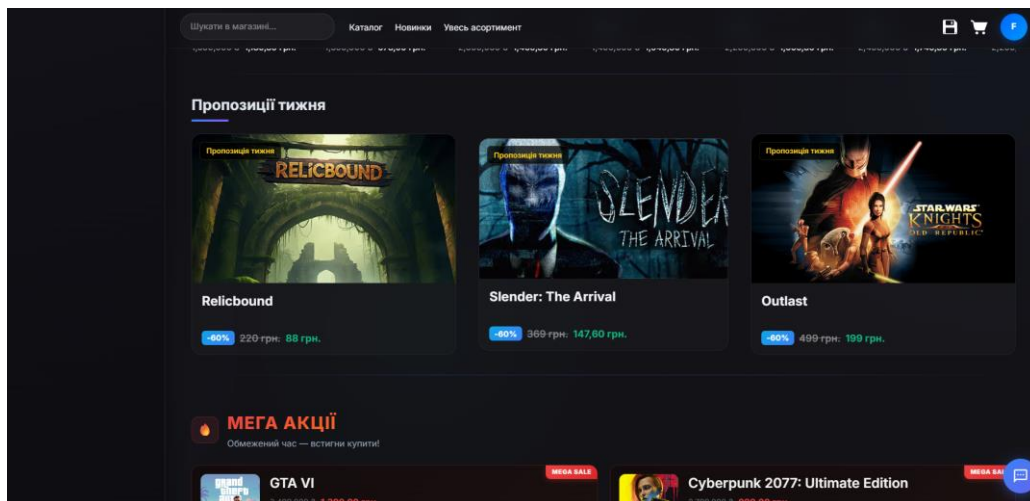
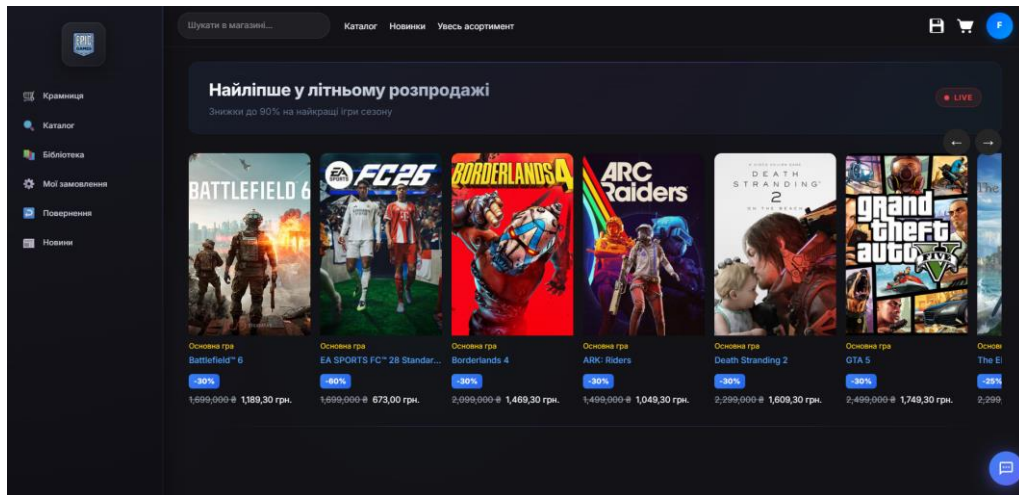
35. Chacon S., Straub B. Pro Git. — 2nd ed. — Apress, 2014. — EPUB (без ISBN). — [Електронний ресурс]. — Режим доступу: <https://git-scm.com/book/en/v2> —

					БР.ІІІ - 07.00.00.000 ПЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

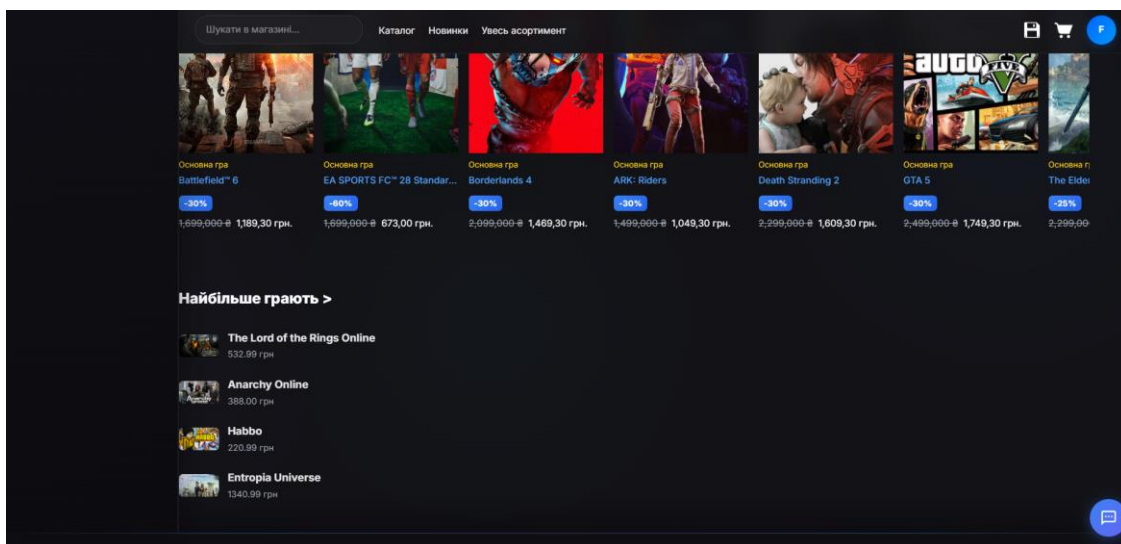
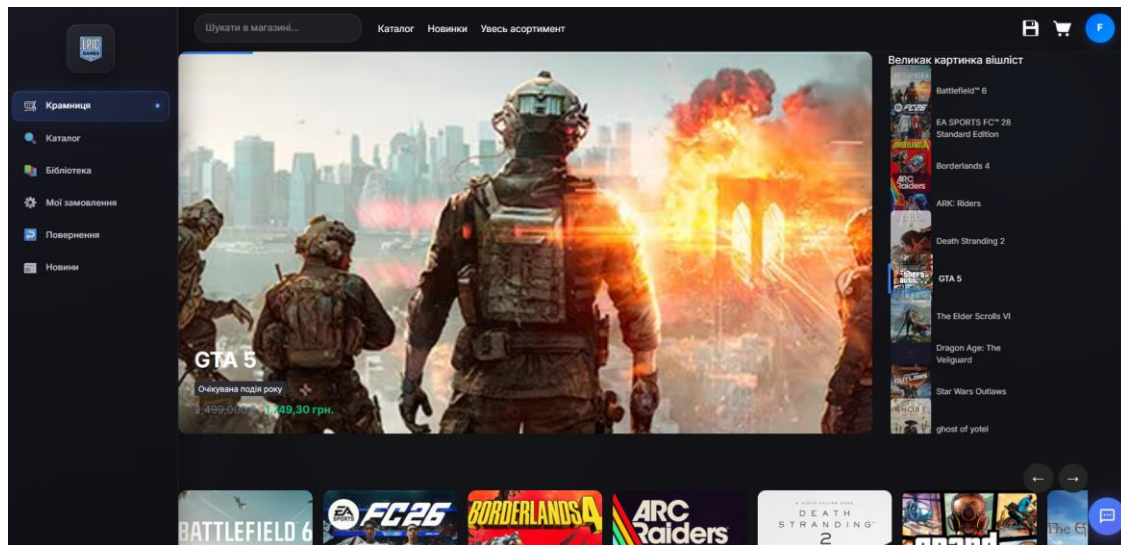
ДОДАТКИ

ДОДАТОК А

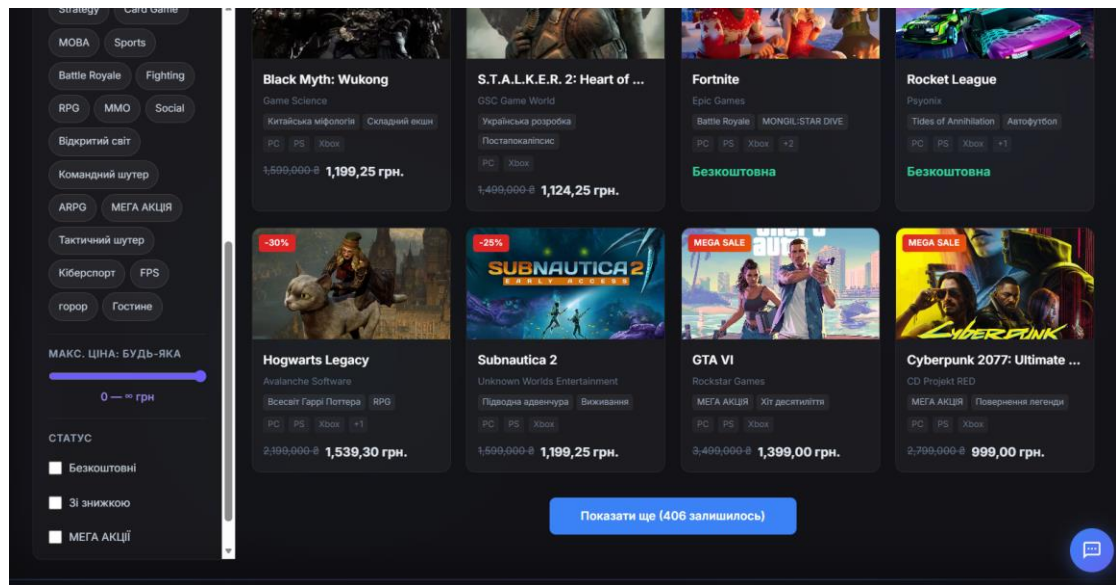
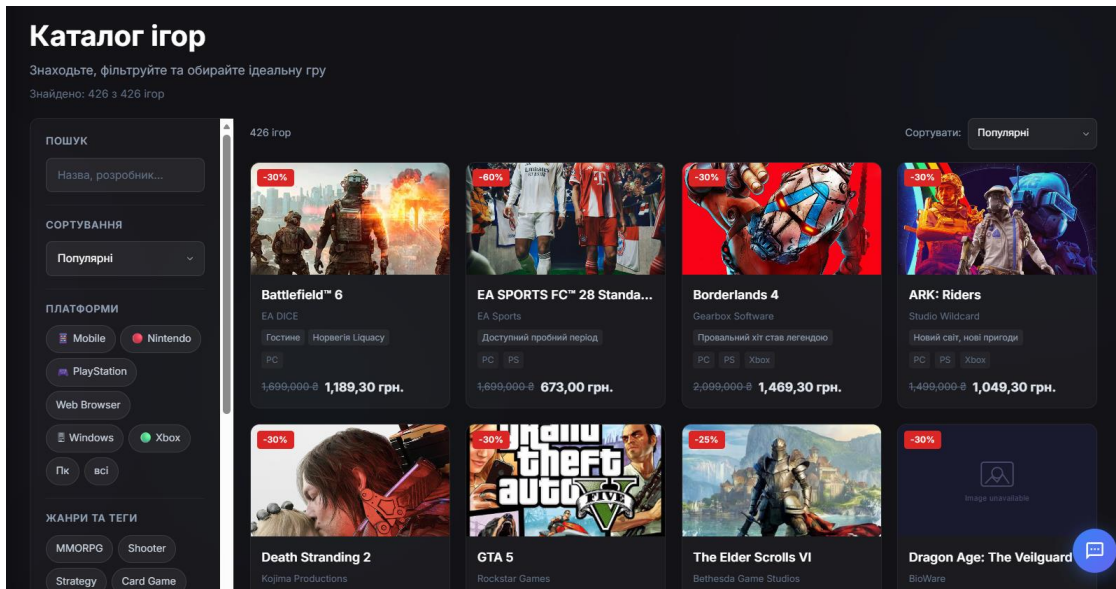
Скріншоти інтерфейсу:



Додаток А1. Демонстрація головної сторінки.



Додаток А2. Демонстрація сторінки крамниці.



Додаток А3. Демонстрація сторінки каталогу з фільтрами.

Увесь асортимент



Battlefield™ 6

Нова частина легендарної серії тактичних шутерів. Battlefield 6

1,189,30 грн. ~~1,699,000 грн.~~



EA SPORTS FC™ 28 Standard Edition

Продовження футбольної симуляції від EA під новим брендом. Покращена

673,00 грн. ~~1,699,000 грн.~~



Borderlands 4

Лутеровий шутер повертається з новими Ваххандами, мільйонами видів

1,469,30 грн. ~~2,099,000 грн.~~



ARK: Riders

Спін-офф серії ARK з акцентом на верхову їзду та повітряні битви на

1,049,30 грн. ~~1,499,000 грн.~~



Death Stranding 2

Екзистенціальна пригода від Хідео Кодзіми розширює межі інтерактивного



GTA 5

Класика відкритого світу з тригерною історією трьох злочинців. Масивний



The Elder Scrolls VI

Довгоочікуване продовження найкращої RPG всіх часів. Дії відбуваються в



Dragon Age: The Veilguard

Повернення до фентезійного всесвіту Тедіс з покращеною бойовою системою

Додаток А4. Демонстрація сторінки всього асортименту.

Головна / Крамниця / ARK: Riders



ARK: Riders

Розробник: Studio Wildcard

Платформи: Windows, PlayStation, Xbox

Теги:

Новий світ, нові пригоди

~~1,499,000 грн.~~ **1,049,30 грн.** **-30%**

Купити гру

Список бажаного

ARK: Riders

Розробник: Studio Wildcard ⌚ 25-40 годин

Новий світ, нові пригоди

ARK: Riders

Розробник: **Studio Wildcard** ⌚ 25-40 годин

Новий світ, нові пригоди

Windows PlayStation Xbox

Спін-офф серії ARK з акцентом на верховну їзду та повітряні битви на драконах. Досліджуйте плавучий архіпелаг, приручайте міфічних істот та беріть участь у масштабних PvP битвах. Для любителів динозаврів та фентезі.



Наша рекомендація

Ця гра варта уваги якщо ви шукаєте унікальний ігровий досвід

Рейтинг гри

0.0

Відгуків: 0

3 10

Відгуки гравців

Поки що немає відгуків. Будьте першим!

Рекомендовані системні вимоги

Minimax

OS version	Win10 64-bits
CPU	Intel i5-8400 or AMD Ryzen5 1500X
Memory	8 GB RAM
GPU	Nvidia GTX1050Ti 4GB or AMD RX580 4GB
DirectX	DirectX 12
Storage	20 GB

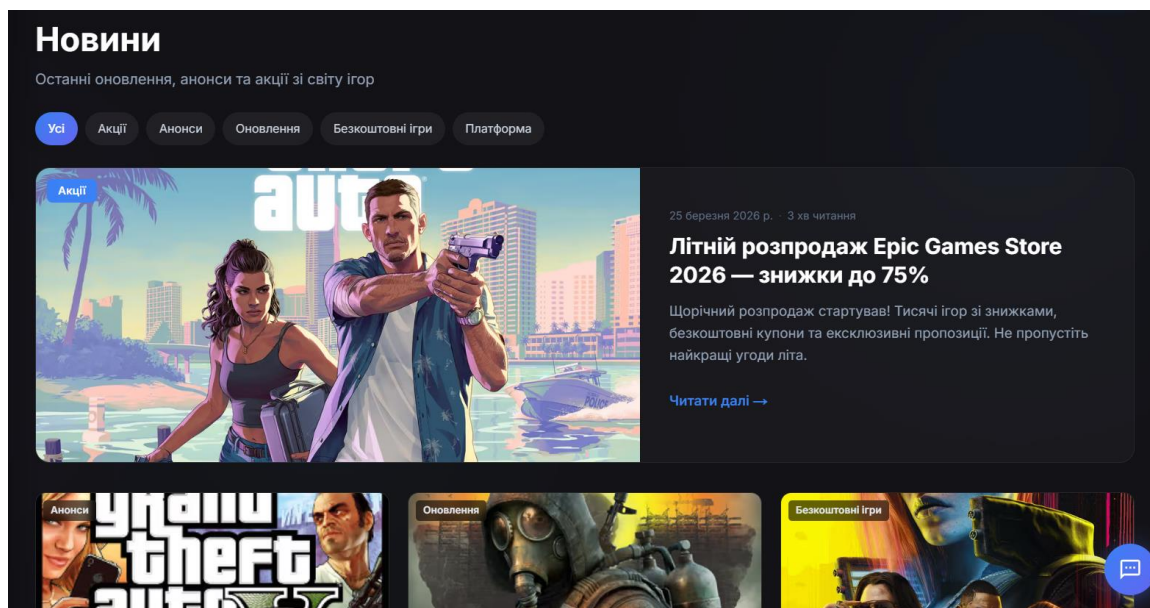
DirectX	DirectX 12
Storage	20 GB
Additional input device	Gamepad

Переглянуті товари

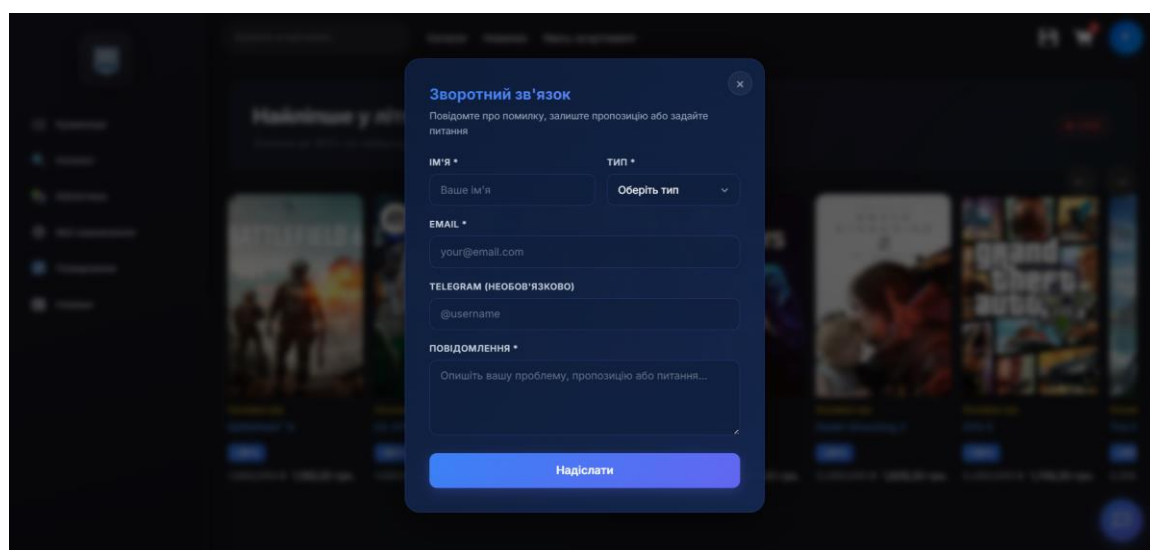
EA SPORTS FC™ 28
Standard Edition

673,00 грн. ~~1,699,000-8~~

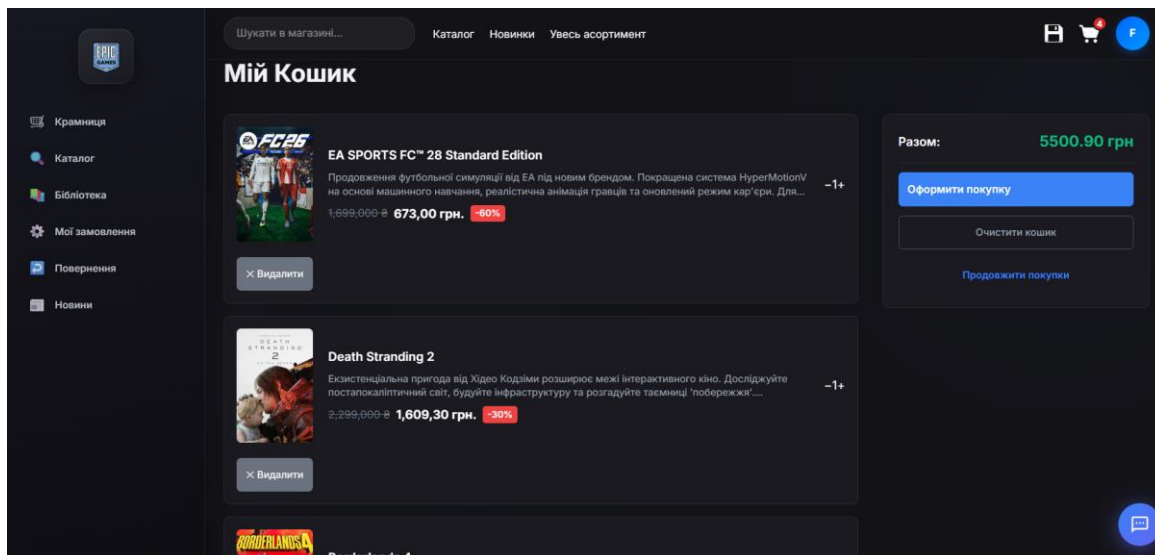
Додаток А5. Демонстрація сторінки карточки товару.



Додаток А6. Демонстрація сторінки актуальних ігрових новин.



Додаток А7. Демонстрація сторінки зворотнього відгуку.



Додаток А8. Демонстрація сторінки кошика.

Оформлення замовлення

Захищено з'єднання (SSL 256-біт)

☐ VISA
 ☐ Mastercard
 ☐ Maestro
 ☐ PayPal
 ☐ Apple Pay
 ☐ Google Pay

Контактні дані

Email для отримання ключів активації

Ключі активації будуть відправлені на цей email

Спосіб оплати

☒ **Кредитна або дебетова картка**
 Visa, Mastercard, Maestro

☐ ПриватБанк / LiqPay
Оплата картою через LiqPay (ПриватБанк)

☐ PayPal
Безпечна оплата через PayPal

Дані карти

ПІДСУМОК ЗАМОВЛЕННЯ

	EA SPORTS FC™ 28 Standard Edition	- 1 +	673.00 грн
	Death Stranding 2	- 1 +	1609.30 грн
	Borderlands 4	- 1 +	1469.30 грн
	GTA 5	- 1 +	
Ціна		5500.90 грн	
ПДВ (20%)		1100.18 грн	
Всього		6601.08 грн	

Промокод

Дані карти

Номер карти

Ім'я на картці

Вкажіть ім'я латиницею, як зазначено на картці

Термін дії

CVV/CVC

CVV/CVC код не зберігається. Усі дані передаються в зашифрованому вигляді.

☐ Я погоджуюсь з [умовами покупки](#)

☐ Зберегти платіжні дані для наступних покупок

ПІДСУМОК ЗАМОВЛЕННЯ

	EA SPORTS FC™ 28 Standard Edition	- 1 +	673.00 грн
	Death Stranding 2	- 1 +	1609.30 грн
	Borderlands 4	- 1 +	1469.30 грн
	GTA 5	- 1 +	
Ціна		5500.90 грн	
ПДВ (20%)		1100.18 грн	
Всього		6601.08 грн	

Промокод

Додаток А9. Демонстрація сторінки оплати.

Дякуємо за покупку!

Номер замовлення: ORD-JYLBVBEO
Дата: 4/10/2026, 12:53:05 AM
Email: fndriy2@gmail.com

✓ Чек з ключами активації відправлено на ваш email!

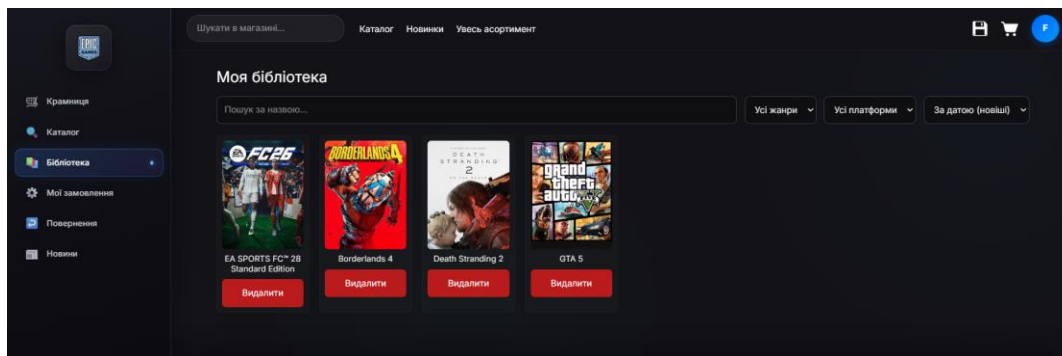
1. EA SPORTS FC™ 28 Standard Edition	1 шт.
2. Death Stranding 2	1 шт.
3. Borderlands 4	1 шт.
4. GTA 5	1 шт.

Ключі активації

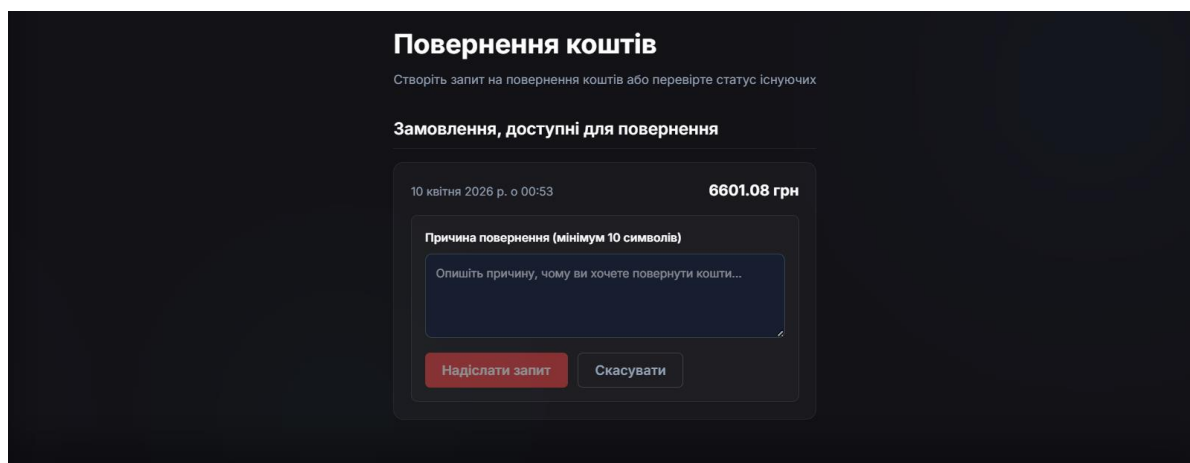
EA SPORTS FC™ 28 Standard Edition	A2B5-FCC5-CCF4-876D
Death Stranding 2	005D-1262-0800-04D9
Borderlands 4	CEC5-2EF4-E932-56C7
GTA 5	C904-70A0-C7C8-A0F6

Всього до сплати: 6601.08 грн

Додаток А10. Демонстрація карточки оплати.



Додаток А11. Демонстрація сторінки бібліотеки акаунту.



Додаток А12. Демонстрація сторінки повернення коштів.



Додаток А13. Демонстрація сторінки моїх замовлень.

ДОДАТОК Б

Таблиці результатів тестування.

№	Назва тесту	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Реєстрація нового користувача	Логін, пароль (2×), роль: customer	Користувача створено, перенаправленн я на головну сторінку	Працює відповідно до очікуваного	PASSED
2	Вхід у систему	Існуючий логін та пароль	Авторизація, перенаправленн я в профіль	Робота коректна	PASSED
3	Вхід з неправильним паролем	Логін існує, пароль невірний	Виведення повідомлення про помилку	Повідомлення відображається	PASSED
4	Перегляд каталогу товарів	-	Відображення всіх товарів	Всі товари відображаються	PASSED
5	Перегляд сторінки товару	Перехід на product/:id	Відображення опису, фото, ціни	Дані показуються правильно	PASSED
6	Додавання товару в кошик	Кнопка “Додати в кошик”	Товар з’являється в кошику	Товар додається успішно	PASSED
7	Видалення товару з кошика	Кнопка “Видалити”	Товар зникає зі списку	Видаляється коректно	PASSED
8	Зміна кількості товару у кошику	+1 / -1	Оновлення суми та кількості	Працює стабільно	PASSED
9	Оформлення замовлення	Дані користувача, підтвердження	Створення замовлення та повідомлення	Замовлення створено	PASSED
10	Перегляд власних замовлень	Увійшовший користувач	Відображення історії замовлень	Історія працює	PASSED
11	Доступ до адмін-панелі без прав	Непривілейований користувач	Заборона доступу	Блокування відбувається	PASSED

1 2	Додавання товару в адмінці	Назва, опис, ціна, фото	Новий товар у каталозі	Працює правильно	PASSED
1 3	Редагування товару в адмінці	Зміна полів	Дані оновлені	Редагування коректне	PASSED
1 4	Видалення товару в адмінці	-	Товар зникає з каталогу	Видається успішно	PASSED

ДОДАТОК В

GitHub - https://github.com/AndriyMykhailiyk/EpicGameShop_V1

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: " Розробка масштабного Full-Stack застосунку для продажу комп'ютерних ігор "

Обсяг пояснювальної записки: 79 аркушів

Дата закінчення бакалаврської роботи “09” червня 2026р.

Підпис студента _____