

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Інтернет-магазин продажу ігор»

Виконав студент групи ІПЗк-21-1

СОКОЛОВ Іван Борисович

Керівник роботи:

КУБРАК Юрій Олександрович

Консультант:

ЄФРЕМОВ Юрій Миколайович

Рецензент:

САВІЦЬКИЙ Роман Святославович

Житомир – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
26 лютого 2024р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **СОКОЛОВ Іван Борисович**

Керівник роботи: КУБРАК Юрій Олександрович

Тема роботи: **«Інтернет-магазин продажу ігор»**, затверджена Наказом закладу вищої освіти №74/с від 23.02.2024р.

Вихідні дані для роботи: розробка інтернет магазин з продажу ігор

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Кубрак Ю.О., Єфремов Ю.М.	27.02.2024р.	27.02.2024р.
2	Кубрак Ю.О., Єфремов Ю.М.	27.02.2024р.	27.02.2024р.
3	Кубрак Ю.О., Єфремов Ю.М.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з сайту продажу ігор та пояснювальної записки.

Текстова частина викладена на 72 сторінках друкованого тексту. Пояснювальна записка має 10 сторінок додатків. Список використаних джерел містить 26 найменувань і займає 2 сторінки. У роботі наведено 44 рисунків та 6 таблиць. Загальний обсяг роботи – 72 сторінок.

У першому розділі було проаналізовано потреби та визначено засоби реалізації.

У другому розділі проведено проектування Інтернет-магазину продажу ігор.

У третьому розділі було розроблено Інтернет-магазин продажу ігор.

Висновок містить в собі результати виконаної роботи при створенні сайту на тему «Інтернет-магазин продажу ігор».

У додатку представлений лістинг розробленого програмного продукту.
Ключові слова: REACT, PRISMA, WEB-САЙТ, MVC, СЕРВЕР, АДМІН, АВТОРИЗАЦІЯ, ДАНІ,

					ІПЗ.КР.Б – 121 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.	І.Соколов				Інтернет-магазин продажу ігор Пояснювальна записка	Літ.	Арк.	Аркушів
Керівник	Ю.Кубрак						3	72
Н. контр.	О.Тичина							
Зав. каф.	Т. Вакалюк					Житомирська політехніка, група ІПЗк-21-1		

ABSTRACT

The graduation thesis of the bachelor consists of an online store selling games and an explanatory note.

The text part is laid out on 72 pages of printed text. The explanatory note has 10 pages of annexes. The list of used sources contains 26 names and occupies 2 pages. The work contains 44 figures and 6 tables. The total volume of work is 72 pages.

In the first section, the needs were analyzed and means of implementation were determined.

In the second section, the design of the online store for the sale of games was carried out.

In the third section, an online store for the sale of games was developed.

The conclusion contains the results of the work done during the creation of the site on the topic "Internet store for the sale of games".

The appendix presents a listing of the developed software product. Keywords: REACT, PRISMA, WEBSITE, MVC, SERVER, ADMIN, AUTHORIZATION, DATA,.

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ІНТЕРНЕТ МАГАЗИНУ ПРОДАЖУ ІГОР	9
1.1Постановка задачі.....	9
1.2АНАЛІЗ АНАЛОГІЧНИХ ПРОГРАМНИХ ПРОДУКТІВ	11
1.3Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	17
Висновки до першого розділу	19
РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ПРОДАЖУ ІГОР	21
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури.....	21
2.2Розробка бази даних системи	27
2.3 Проектування та реалізація алгоритмів роботи інтернет-магазину	30
2.4 Реалізація функціоналу інтернет магазину з продажу ігор	32
Висновки до другого розділу	40
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ МАГАЗИНОМ ПРОДАЖУ ІГОР	41
3.1 Порядок встановлення та налаштування параметрів інтернет магазину	41
3.2 Структура інтерфейсу інтернет магазину продажу ігор	42
3.3 Тестування роботи інтернет магазину з продажу ігор	55
Висновки до розділу 3	57
ВИСНОВКИ.....	58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	60
ДОДАТКИ.....	62

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

CMS – система управління контентом сайту.

JS – JavaScript

MVC – Модель–вигляд–контролер (або Модель–представлення–контролер, Model-view-controller, MVC) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

БД – База даних.

ПЗ – Програмне забезпечення.

ПК – персональний комп’ютер.

СУБД – система управління базами даних.

ВСТУП

Актуальність теми: Сучасний ринок відеоігор є одним із найдинамічніших і найприбутковіших сегментів індустрії розваг. В останні десятиліття відеоігри стали не лише засобом відпочинку та розваги, а й важливим культурним явищем, що приваблює мільйони геймерів у всьому світі. У цьому контексті розвиток інтернет-магазинів з продажу ігор має велике значення з кількох причин. Зростання попиту на відеоігри: Кількість геймерів зростає з кожним роком, що призводить до стабільного попиту на нові ігри. Інтернет-магазини можуть задовольнити цей попит, пропонуючи широкий асортимент ігор для різних платформ і вікових груп. Зручність для споживачів: Інтернет-магазини пропонують можливість придбати ігри в будь-який час і в будь-якому місці. Це особливо важливо в сучасному ритмі життя, коли люди цінують свій час і шукають найзручніші способи покупки. Розширення ринку збуту: онлайн-платформи дозволяють розширити географічні межі компанії та охопити клієнтів у різних частинах світу. Це створює нові можливості для збільшення продажів і підвищення прибутковості. Тенденція до цифрового розповсюдження: Все більше гравців віддають перевагу цифровим копіям ігор через зручність і доступність. Інтернет-магазини можуть миттєво доставляти цифрові версії ігор, що дозволяє швидко реагувати на попит.

Конкурентоспроможність: В умовах жорсткої конкуренції на ринку відеоігор наявність інтернет-магазину є необхідною умовою для виживання та розвитку компанії.

Враховуючи вищезазначені фактори, тема створення та розвитку інтернет-магазину ігор є дуже актуальною. Він поєднує в собі як економічні, так і технічні аспекти, дозволяючи дізнатися не тільки про процеси торгівлі, а й про вплив новітніх технологій на сучасний бізнес. Таким чином, це дослідження сприяє розумінню основних тенденцій на ринку відеоігор і допомагає розробити ефективні стратегії успіху бізнесу в цій галузі.

Мета даної роботи полягає розробити та продемонструвати ефективну модель інтернет-магазину з продажу відеоігор, яка забезпечує високу

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						7

конкурентоспроможність на ринку, задовольняє потреби клієнтів та сприяє стабільному розвитку бізнесу.

Встановлена мета обумовлює наступні завдання:

- аналіз ринку відеоігор;
- вивчення потреб споживачів;
- розробка структури інтернет-магазину;
- оцінка технологічних рішень;
- створення інтерфейсу;
- аналіз існуючих аналогів;
- оцінка фінансової ефективності;
- впровадження та тестування

Об’єктом дослідження є методи та засоби розробки інтернет магазину в рамках автоматизації обробки даних і створення зручного та швидкодіючого веб-сайту, який підвищує задоволеність клієнтів.

Предметом дослідження є використання технологій React для розробки сайту з продажу відеоігор.

У роботі були застосовані методи системного аналізу, функціонального підходу, об’єктно-орієнтованого проектування та дизайну користувацьких інтерфейсів.

Розроблений інтернет магазин може бути використаний на всіх платформах починаючи від телефону і закінчуючи комп’ютером. Через свою гарну оптимізацію.

Випускна кваліфікаційна робота складається з вступу, 3-х розділів, висновків, списку використаних джерел із 26 найменувань (2 стор.), 4 додатків (10 стор.), 6 таблиць та 44 ілюстрацій по тексту. Повний обсяг роботи становить 72 сторінки, з них 65 сторінки основного тексту.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ІНТЕРНЕТ МАГАЗИНУ ПРОДАЖУ ІГОР

1.1 Постановка задачі

Метою кваліфікаційної роботи бакалавра є розробка інтернет-магазину для продажу ігор з використанням новітніх технологій, щоб забезпечити простоту використання та збільшити обсяг продажів у майбутньому.

Інтернет-магазин, який продає ігри, повинен містити головну сторінку, каталог товарів, кошик, сторінку входу та сторінку реєстрації. З боку адміністратора необхідно забезпечити можливість додавання, редагування та видалення товарів, відстеження продажів і користувачів, а також ведення статистики.

Щоб реалізувати цей програмний продукт, необхідно використовувати локальний сервер, наприклад локальний сервер, вбудований у Visual Studio Code. Для виконання дипломної роботи слід застосувати такі мови програмування:

- HTML для створення структури веб-сторінок;
- CSS для розробки стилю (зовнішнього вигляду) сторінки;
- JavaScript як мова сценаріїв для додавання інтерактивності веб-сторінок, а також його фреймворк Next.js для розробки серверної частини та React.js для функціональної частини [1, 2].

Для успішної втілення проекту в життя необхідно використовувати об'єктно-орієнтоване програмування. Цей підхід до програмування спрямований на моделювання інформаційних об'єктів, що спрощує управління програмними процесами та поліпшує структурування інформації. Використання об'єктно-орієнтованого програмування допомагає підвищити зрозумілість коду, спростити роботу з ним, зробити програмний продукт більш структурованим і допомагає уникнути деяких типових помилок, що виникають під час розробки.

З огляду на вище сказано для проекту було складений перелік задач які потрібно вирішити задля досягнення даного результату:

- Провести аналіз ринку інтернет-магазинів відеоігор;
- Виявлення основних вимог користувачів до такого магазину;
- Дослідити існуючі рішення та їхні недоліки;
- Розробка архітектури інтернет-магазину;
- Визначте функціональні та нефункціональні вимоги до системи;
- Визначити відповідні технології для розробки фронтенду та бекенду;
- Обрати базу даних для зберігання інформації про користувачів, ігри та транзакції.
- Визначити засоби для забезпечення безпеки даних та захисту від зломів.
- Інтегрувати платіжні системи для здійснення транзакцій.
- Провести тестування функціональності системи.
- Оптимізувати продуктивність та швидкість завантаження сторінок.
- Забезпечити кросбраузерну сумісність та адаптивність для мобільних пристроїв.

Для втілення зазначених завдань у системі потрібно мати такі об'єкти:

1. Каталог ігор:
 - 1.1. Сортування за жанрами.
 - 1.2. Пошук за назвою.
 - 1.3. Відображення опису, зображень та інформації про гру.
2. Кошик та замовлення:
 - 2.1. Додавання товарів до кошика.
 - 2.2. Оформлення замовлення з вибором способу та оплати.
3. Акції та знижки:
 - 3.1.Проведення розпродажів, знижок на певні ігри.
4. Особистий кабінет користувача:
 - 4.1. Реєстрація/вхід у свій акаунт.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						10

- 4.2. Перегляд історії замовлень.
5. Система рецензій та оцінок:
 - 5.1. Можливість користувачів залишати відгуки та оцінки для ігор.
 - 5.2. Відображення середнього рейтингу гри на сторінці товару.
6. Підтримка платіжних систем:
 - 6.1. Інтеграція з різними платіжними системами.
7. Адміністративна панель:
 - 7.1. Управління каталогом товарів (додавання, редагування, видалення).
 - 7.2. Керування замовленнями та клієнтською базою.
 - 7.3. Статистика продажів, аналітика роботи магазину.
8. Підтримка мобільних пристроїв:
 - 8.1. Адаптивний дизайн для зручного перегляду на смартфонах та планшетах.
9. Соціальні мережі:
 - 9.1. Можливість обміну даними про покупки або відгуки через соціальні мережі.
10. Безпека та конфіденційність:
 - 10.1. Захист особистих даних користувачів, шифрування платіжних даних.

1.2 Аналіз аналогічних програмних продуктів

Для проведення порівняльного аналізу були обрані наступні п'ять інтернет-магазинів з продажу ігор: Keybox, Mgames, Gameshop, Retromagaz та I-gamer. Аналіз цих магазинів може дати розуміння застосованих практик і підходів, що в подальшому може позитивно вплинути на розробку власного веб-додатка. Тому проаналізуємо кожен із цих сайтів:

1. Keybox [3].

Посилання на веб-сайт: <https://keybox.com.ua/>

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						11

Вигляд сайту – рис 1.1

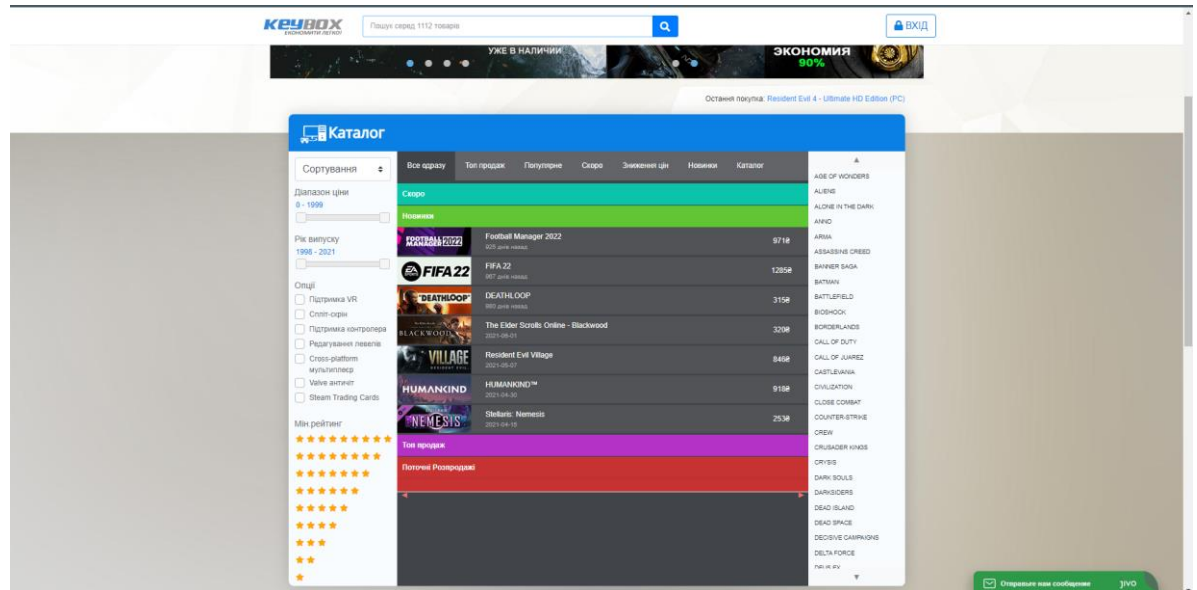


Рис 1.1 Інтернет магазин Keybox

Даний сайт є одним з найдавніше створених базою продажу ігор для приставок, через що візуальна частина може бажати кращого, але й проста у використанні.

Основні переваги які можна зазначити:

- швидке завантажування сторінок;
- проста функціональна частина для клієнта;
- веб-додаток має підтримку різними мовами;
- великий каталог ігор, навіть старих яких важко знайти;
- цілодобова підтримка споживача;
- реальні відгуки клієнтів;

З недоліків підкреслити можна наступні фактори:

- застарілий інтерфейс;
- неоптимізований сайт під різні платформи;

2. Mgames [4].

Посилання на веб-сайт: <https://mgames.ua/>

Вигляд сайту – рис 1.2

3. Gameshop [5].

Посилання на веб-сайт: <https://game-shop.com.ua/ua>

Вигляд сайту – рис 1.3

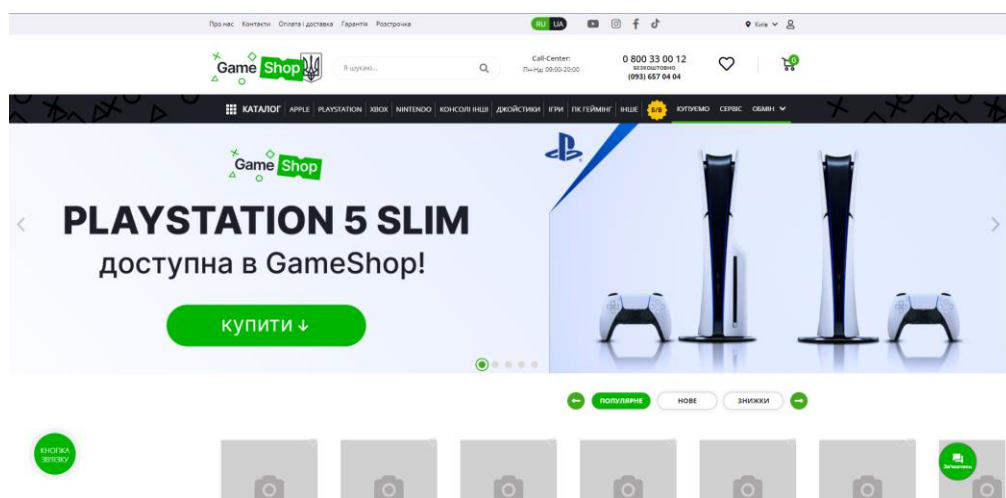


Рис 1.3 Інтернет магазин Gameshop

Gameshop є онлайн-магазином, що спеціалізується на продажу ігрових консолей, ігор та аксесуарів. Вебсайт пропонує широкий асортимент товарів для різних платформ, включаючи PlayStation, Xbox, Nintendo, а також старі консолі. Додатково можна придбати продукцію Apple, аксесуари для ПК-геймінгу, настільні ігри та інші товари для розваг. Магазин також продає вживані товари, що дозволяє купувати ігри та консолі за зниженими цінами. На сайті доступні розділи для акцій, обміну, сервісу та прокату обладнання.

Основні переваги які можна зазначити:

- сучасний приємний дизайн;
- гаряча лінія без вихідних;
- доставка в різні міста країни;
- клієнто орієнтованість;

З недоліків підкреслити можна наступні фактори:

- затримка доставки;
- високі ціни;

4. Retromagaz [6].

Посилання на веб-сайт: <https://retromagaz.com/>

Вигляд сайту – рис 1.4

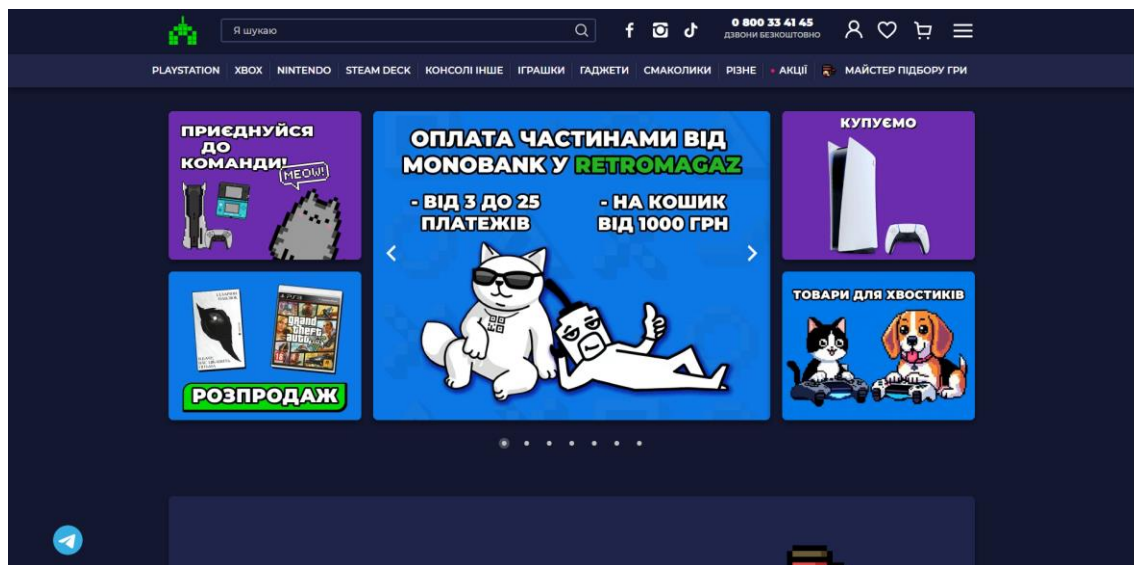


Рис 1.4 Вигляд сайту Retromagaz

Сайт RetroMagaz спеціалізується на продажу ігрових консолей, ігор та аксесуарів. Він пропонує широкий асортимент як сучасних консолей (PlayStation 5, Xbox Series X|S, Nintendo Switch), так і ретро-консолей (Sega, Dendy, GameBoy). Крім того, магазин має великий вибір аксесуарів, іграшок, гаджетів і навіть смаколиків. На сайті також можна знайти ексклюзивні та рідкісні видання консолей. RetroMagaz надає можливість попереднього замовлення та сповіщення про наявність товарів. Сайт має зручну навігацію та різноманітні розділи для легкого пошуку товарів.

Основні переваги які можна зазначити:

- широкий асортимент;
- зручна навігація;
- ексклюзивні товари;
- попереднє замовлення;
- сповіщення про наявність;

З недоліків підкреслити можна наступні фактори:

- обмежена мовна підтримка;
- відсутність міжнародної доставки;
- обмежений асортимент для деяких платформ;
- цінова політика;

- технічні збої;

5. I-gamer [7].

Посилання на веб-сайт: <https://i-gamer.ua/>

Вигляд сайту – рис 1.5

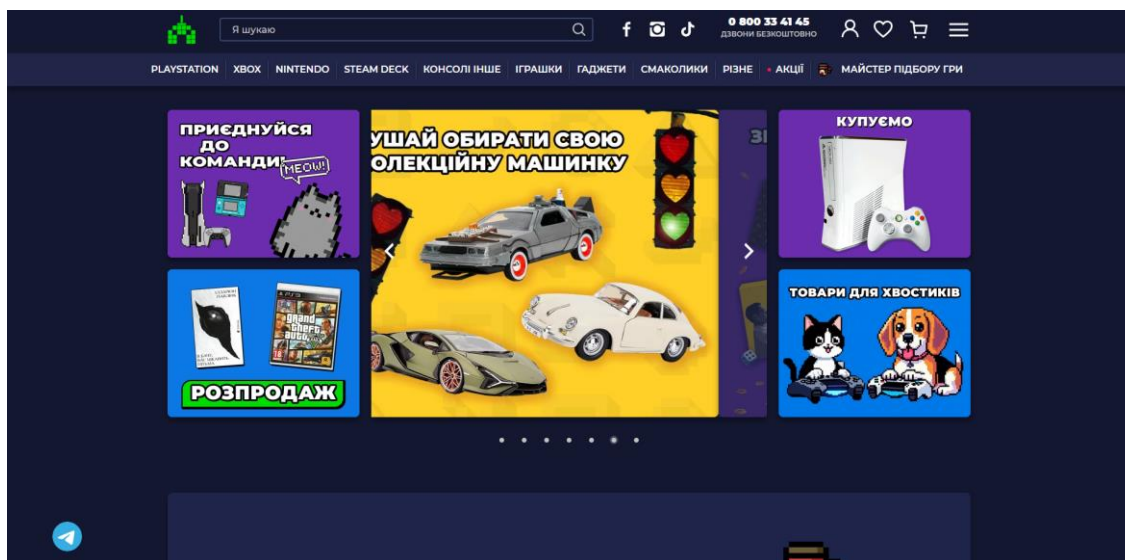


Рис 1.5 Вигляд сайту I-gamer

i-Gamer.ua є інтернет-магазином, що спеціалізується на продажу ігрових консолей, ігор, аксесуарів та геймерської атрибутики. Сайт пропонує товари від провідних брендів, таких як Sony, Microsoft, Nintendo та інших. Він має добре організований каталог товарів, що включає.

Основні переваги які можна зазначити:

- широкий асортимент;
- зручна структура каталогу;
- брендові товари;
- методи оплати і доставки;
- акції та спеціальні пропозиції;

З недоліків підкреслити можна наступні фактори:

- мовна підтримка;
- обмежена міжнародна доставка;
- технічні проблеми;
- обмежений асортимент для деяких платформ;

Наведено порівняння характеристик інтернет-магазинів: Keybox, Mgames, Gameshop, Retromagaz, I-gamer, а також розроблюваного власного проекту в рамках кваліфікаційної роботи (табл. 1.1).

Таблиця 1.1

Характеристика	Keybox	Mgames	Gameshop	Retromagaz	I-gamer	Розроблюваний проект
Ціни	Низькі	Високі	Високі	Низькі	Високі	Низькі
Функціонал	Основний	Розширений	Розширений	Розширений	Основний	Основний
Інтерфейс	Застарілий	Сучасний	Сучасний	Сучасний	Простий	Простий, мінімалістичний
Доставка	Відсутня	Є	Є	Є	Є	Є
Гаряча лінія	Відсутня	Є	Є	Є	Є	Відсутня

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Веб-додаток буде побудований на клієнт-серверній архітектурі. Для клієнтської частини буде використано фреймворк Next.js, заснований на бібліотеці React.js [8-12]. Для функціоналу також використовується мова програмування Typescript та для роботи з візуалом тексту та сторінок Tailwind. В якості сховища даних буде використовуватися база даних MongoDB [13, 14].

Next.js — популярний фреймворк для створення веб-додатків на основі React.js, розроблений компанією Vercel. Він надає розширені функції для створення сучасних веб-додатків, включаючи рендеринг на стороні сервера, генерацію статичного сайту та інші функції, які спрощують розробку високопродуктивних масштабованих програм.

Переваги Next.js:

Серверний рендеринг (Server-Side Rendering, SSR): Покращена продуктивність сторінки, відображаються на сервері перед надсиланням клієнту, що призводить до швидшого часу завантаження також SEO-оптимізація Пошукові системи можуть індексувати повністю відтворені сторінки, підвищуючи видимість веб-сайту.

Генерація статичних сайтів (Static Site Generation, SSG): Швидкість: сторінки генеруються під час створення, гарантовано миттєво завантажуються. Безпека: Статичні сайти не вразливі до атак, оскільки вони не мають динамічних компонентів на сервері.

Автоматичний розподіл коду (Automatic Code Splitting): Ефективність завантаження: Код ділиться на окремі частини, що зменшує обсяг завантажуваних даних і покращує швидкість роботи додатка.

API маршрути (API Routes): Інтегровані серверні функції: Дозволяє створювати серверні API в рамках одного проекту, спрощуючи архітектуру додатка.

TypeScript – це мова програмування, розроблена компанією Microsoft, яка є надбудовою над JavaScript. Вона додає статичну типізацію до мови JavaScript, що дозволяє виявляти помилки на етапі компіляції, а не під час виконання програми. TypeScript є відкритим кодом і має велику спільноту розробників [15-17].

Переваги використання TypeScript:

Зменшення кількості помилок: Статична типізація допомагає виявляти помилки на етапі компіляції, що зменшує кількість багів в робочому коді.

Покращення читабельності та підтримки коду: Чітке визначення типів та інтерфейсів робить код більш зрозумілим та легшим для підтримки, особливо у великих проектах.

Інтеграція з сучасними фреймворками: TypeScript активно використовується у багатьох сучасних фронтенд- і бекенд-фреймворках, таких як Angular, React, Vue, Node.js тощо.

Підвищення продуктивності розробників:

Автокомплектція, рефакторинг та інші інструменти значно прискорюють процес розробки та покращують якість коду.

Tailwind CSS – це утилітарний CSS-фреймворк, який надає набір низькорівневих класів для створення користувацьких інтерфейсів безпосередньо в HTML. Він відрізняється від традиційних CSS-фреймворків, таких як Bootstrap, тим, що не включає попередньо розроблені компоненти, а натомість пропонує утилітарні класи, які можна комбінувати для створення складних конструкцій [18].

Переваги використання Tailwind CSS:

Швидкість розробки: Швидке створення прототипів: Використання утилітарних класів дозволяє швидко створювати прототипи та перевіряти дизайн-ідеї. Мінімальна потреба в написанні кастомного CSS: Більшість стилів можна реалізувати за допомогою готових класів, що зменшує обсяг кастомного CSS-коду.

Консистентність дизайну: Єдина дизайн-система: Використання утилітарних класів забезпечує консистентність усього інтерфейсу. Легкість підтримки: Чітка структура класів полегшує підтримку і внесення змін до стилів.

Гнучкість та налаштування: Конфігуровані теми: Можливість налаштовувати кольори, шрифти, відступи та інші параметри через конфігураційний файл. Підтримка плагінів: Можливість додавання кастомних утилітарних класів та плагінів для розширення функціональності Tailwind CSS.

Адаптивний дизайн: Медіа-запити: Вбудована підтримка адаптивного дизайну з використанням медіа-запитів для різних розмірів екранів. Респонсивні утиліти: Класи для адаптивного дизайну дозволяють швидко змінювати стилі залежно від розміру екрана.

Висновки до першого розділу

У першому розділі було проведено детальний аналіз напрямків застосування інтернет-магазину продажу ігор. Поставлена задача, визначено

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						19

основні цілі та завдання, що стоять перед розробкою даного проекту, зокрема, створення ефективної платформи для продажу ігор, яка задовольняє потреби користувачів та забезпечує високий рівень продуктивності і зручності.

Аналіз аналогічних програмних продуктів дозволив виявити сильні та слабкі сторони існуючих рішень на ринку. Це дало можливість врахувати найкращі практики та уникнути типових помилок при розробці власного проекту. В результаті цього аналізу були визначені ключові функціональні можливості та вимоги до інтернет-магазину, що включають зручний інтерфейс користувача, безпечну систему оплати, ефективне управління контентом та високий рівень продуктивності.

Обґрунтування вибору інструментальних засобів базувалося на аналізі сучасних технологій, що дозволяють забезпечити стабільну роботу та розвиток інтернет-магазину. Було прийнято рішення використовувати Next.js як основний фреймворк для клієнтської частини, TypeScript для забезпечення надійності коду, Tailwind CSS для швидкої розробки інтерфейсу та MongoDB як базу даних для зберігання інформації. Це рішення дозволяє досягти високої продуктивності, масштабованості та зручності підтримки проекту.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ПРОДАЖУ ІГОР

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури

У цьому розділі проведено аналіз вимог для створення власного інтернет-магазину з продажу ігор. Він передбачає детальне вивчення потреб та очікувань користувачів щодо функціональності та характеристик. Основною метою аналізу вимог є визначення того, що потрібно реалізувати в програмному забезпеченні для задоволення потреб користувачів. Завдяки інтерв'ю з користувачами та тестерами продукту, опитуванням та зворотному зв'язку були визначені функціональні вимоги, які окреслюють специфічні особливості та можливості, які повинен мати веб-застосунок.

Додатково були зібрані нефункціональні вимоги, що стосуються аспектів продуктивності, стабільності роботи та зручності використання. Цей магазин дозволить покупцям зручно переглядати товари, підбирати їх за категоріями, переходити до оплати та залишати відгуки після покупки. Користувачі будуть поділені на три групи: неавторизовані користувачі, авторизовані користувачі та адміністратори. Для наочного відображення взаємодії користувачів з сайтом буде створено діаграму варіанту роботи сайту та загального його потрібності (рис. 2.1).

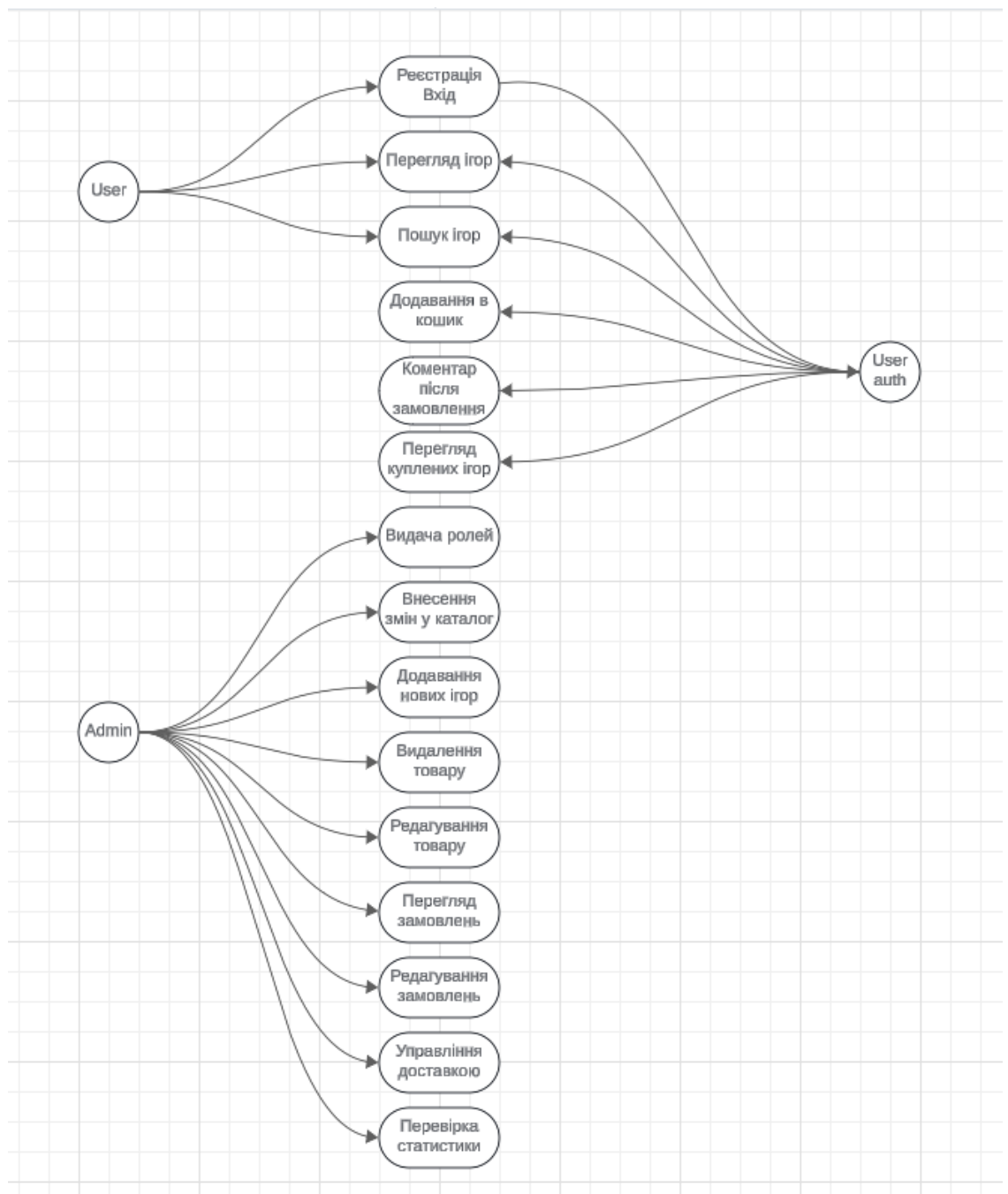


Рис 2.1 Діаграма функціональної потреби сайту

Головною ціллю розробки інтернет магазину з продажу ігор створити онлайн-середовище, де нові ігри можна замовляти та доставляти в будь-яку точку світу. Також є прагнення надати користувачам простий у користуванні та надійний веб-сайт, щоб користувачі могли залишати відгуки, оцінювати продукти та висловлювати свою думку.

Основною метою роботи на тему «Інтернет-магазин з продажу ігор» є розробка та впровадження сучасного, зручного та функціонального веб-

додатку, який гарантує ефективний процес продажу цифрових ігор. Це включає створення інтуїтивно зрозумілого інтерфейсу, надання безпечних методів оплати, інтеграцію з популярними платформами розповсюдження ігор та впровадження системи адміністративного керування для управління продуктами, замовленнями та користувачами.

З цього можна підвести підсумок початкових умов та вимог для користувачів:

Адміністратор:

1. Додавання, редагування, видалення товару;
2. Перегляд статистики замовлень;
3. Видача ролей;
4. Управління доставкою;

Авторизований користувач:

1. Авторизація
2. Пошук товару
3. Перегляд товару
4. Перегляд кошика
5. Можливість купівлі
6. Перегляд історії купівель

Неавторизований користувач:

1. Перегляд товару
2. Пошук товару

Наступним фактором стане вимоги до сайту. Для більш детального ознайомлення представимо їх за списком:

Функціональні вимоги:

1. Інтуїтивно зрозумілий інтерфейс: розробити зручний для користувачів інтерфейс, що дозволяє легко шукати, переглядати та купувати ігри.
2. Система безпечних платежів: забезпечити надійні та безпечні методи оплати, підтримуючи різноманітні платіжні системи.

3. Адміністративна панель: розробити систему адміністрування для управління товарами, замовленнями, користувачами та іншими аспектами магазину.
4. Особистий кабінет користувача: забезпечити користувачам можливість створювати та управляти своїми обліковими записами, переглядати історію покупок та інше.
5. Пошук та фільтрація: реалізувати функції пошуку ігор за критеріями такими як жанр.
6. Оцінки та відгуки: впровадити систему оцінок і відгуків, яка дозволить користувачам ділитися своїми враженнями від куплених ігор.

Нефункціональні вимоги:

1. Безпека: Забезпечити високий рівень безпеки даних користувачів, включаючи захист персональної інформації та платіжних даних від несанкціонованого доступу.
2. Продуктивність: Система повинна обробляти запити користувачів швидко, забезпечуючи мінімальний час завантаження сторінок і високий рівень відповіді на запити.
3. Масштабованість: Платформа повинна бути здатною масштабуватися для підтримки збільшення кількості користувачів та транзакцій без втрати продуктивності.
4. Надійність: Забезпечити високу надійність роботи системи, мінімізуючи кількість збоїв та простоїв.
5. Юзабіліті: Інтерфейс має бути інтуїтивно зрозумілим і легким у використанні для користувачів різних вікових груп і технічних навичок.
6. Сумісність: Забезпечити сумісність веб-додатку з різними браузерами та пристроями, включаючи настільні комп'ютери, планшети та смартфони.

7. Обслуговування та підтримка: Система повинна мати можливість легкого обслуговування та оновлення, включаючи додавання нових функцій та виправлення помилок.
8. Масив даних: Оптимізувати роботу з великими обсягами даних, забезпечуючи ефективне зберігання, пошук та обробку інформації про користувачів, ігри та транзакції.

Системні вимоги:

Вимоги до середовища виконання:

1. Мінімальні апаратні вимоги для клієнтських пристроїв:
 - Оперативна пам'ять: Не менше 3 ГБ.
 - Процесор: Тактова частота не нижче 1.5 GHz.
 - Інтернет-з'єднання: Постійний доступ до мережі Інтернет.

Вимоги до серверу інформаційної системи:

2. База даних:
 - Тип: Документаційно орієнтована база даних MongoDB.
3. Веб-додаток:
 - Технології: Орієнтація на Web 3.0 технології
4. Підтримка NodeJS:
 - Серверні технології: Сервер повинен підтримувати NodeJS для забезпечення функціональності веб-додатку [19].

За допомогою аналізу функціональних вимог з'явилась можливість визначити наступні сутності, які забезпечать програмну реалізацію додатку. На рисунку 2.2 представлено діаграму класів, що використовуються для його функціонування.

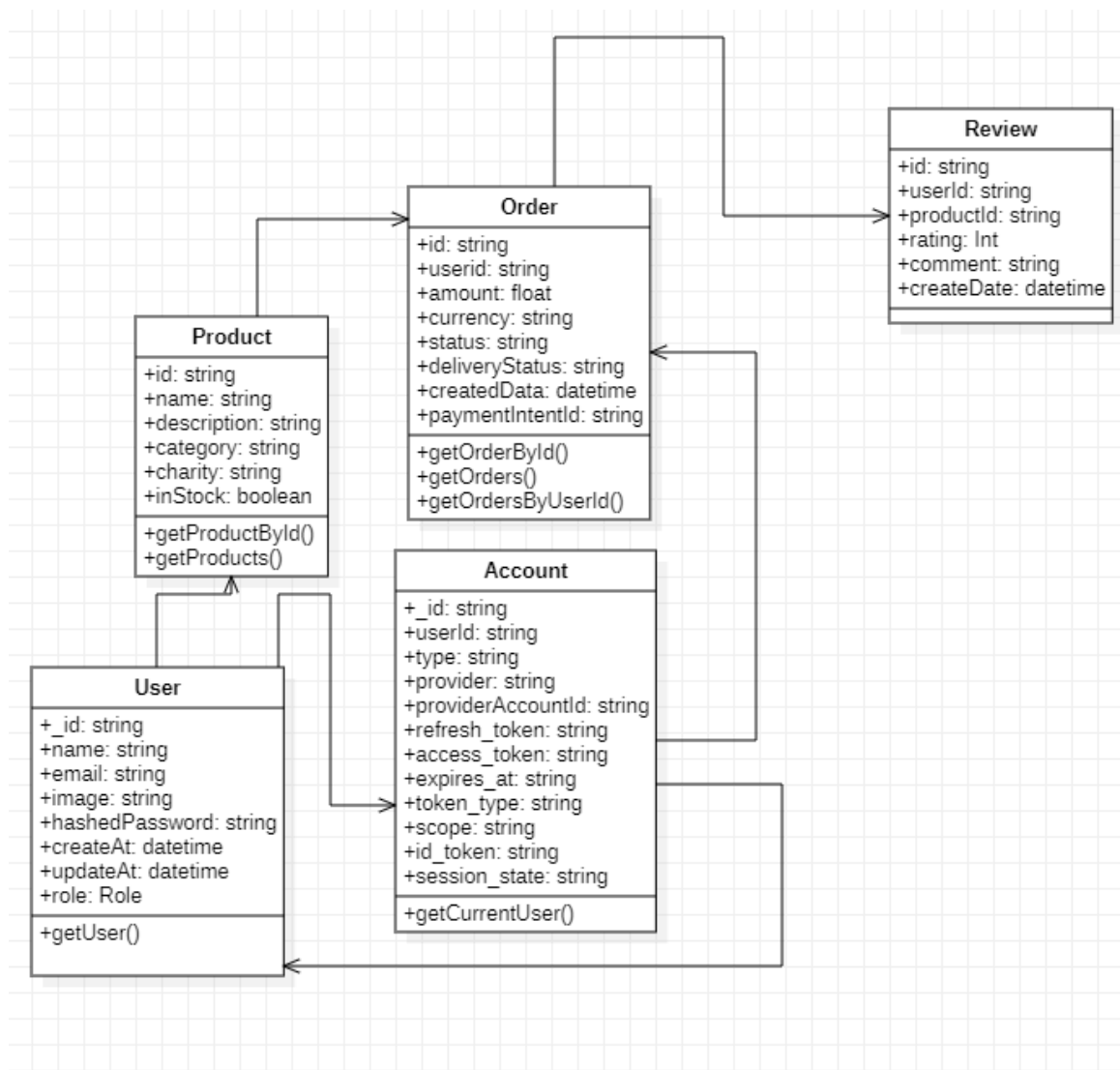


Рис 2.2 Діаграма класів

User - клас, що відповідає за користувача:

- `getUser()` – відповідає за реєстрацію користувача;

Account – клас, що відповідає за авторизацію користувача через Google [20]:

- `getCurrentUser()` – перевірка наявного користувача після створення;

Product – клас, що відповідає за товар:

- `getProducts()` – відповідає за повний список товарів;
- `getProductById()` – відповідає за окремо відкритий товар;

Order – клас, що відповідає за замовлення:

- `getOrderById()` – відповідає за пошук замовлення по його id;

- `getOrders()` – відповідає за список всіх замовлень які були створені;
- `getOrdersById()` – відповідає за створення замовлення користувачем;

`Review` – клас, що відповідає за коментарі після замовлення;

2.2 Розробка бази даних системи

Для зберігання інформації у інтернет магазині буде використовуватись MongoDB. Вибір на неї впав через гнучкість схеми та високу продуктивність.

База даних інтернет магазину складається з п'яти таблиць:

1. `Account` – ця таблиця відповідає за зберігання інформації авторизованих користувачів через Google.
2. `Order` – ця таблиця відповідає за замовлення
3. `Product` – ця таблиця відповідає за товар магазину
4. `Review` – ця таблиця відповідає за коментарі в замовленнях
5. `User` – ця таблиця відповідає за всіх користувачів інтернет магазину

Таблиця 2.1

Опис таблиці `Account`

Назва	Тип	Роль
<code>_id</code>	string	Ідентифікатор
<code>userId</code>	string	Ідентифікатор користувача
<code>type</code>	string	тип
<code>provider</code>	string	Використовувана для авторизації соц. мережа
<code>providerAccountId</code>	string	Ідентифікатор користувача на стороні соц. мережі
<code>access_token</code>	string	Унікальний токен

		користувача
expires_at	int	Термін дії токена
token_type	string	Тип токена
scope	string	Посилання на токен для авторизації
id_token	string	Унікальний ідентифікатор токена користувача

Таблиця 2.2

Опис таблиці Order

Назва	Тип	Роль
_id	string	Ідентифікатор
userId	string	Ідентифікатор користувача
amount	int	Ціна замовлення
currency	string	Валюта в якій йде оплата
status	string	Статус оплати від клієнта
deliveryStatus	string	Статус доставки
createdData	dateTime	Час коли було створенно замовлення
paymentIntentId	string	Унікальний ідентифікатор замовлення
products	string	Товар який купується

Таблиця 2.3

Опис таблиці Product

Назва	Тип	Роль
_id	string	Ідентифікатор
name	string	Назва товару
description	string	Опис товару
price	float	Ціна товару
brand	string	Бренд який випускає товар
category	string	Категорія
inStock	boolean	наявність
images	string	картинка

Таблиця 2.4

Опис таблиці Review

Назва	Тип	Роль
_id	string	Ідентифікатор
userId	string	Ідентифікатор користувача
productId	string	Ідентифікатор продукту
rating	int	Рейтинг товару
comment	string	Коментар залишений після купівлі
createDate	dateTime	Час коли був створений коментар

Опис таблиці User

Назва	Тип	Роль
_id	string	Ідентифікатор
name	string	Ім'я клієнта
Email	string	Електронна адреса користувача
hashedPassword	string	Зашифрований пароль
createdAt	dateTime	Час коли був створений пароль
updatedAt	dateTime	Час коли останній раз його оновили
Role	string	Роль користувача на сайті

2.3 Проектування та реалізація алгоритмів роботи інтернет-магазину

Додаток є простим у використанні, має всі основні функції інтернет-магазину, що дозволяє користувачам легко взаємодіяти з ним. Алгоритм роботи з інтернет-магазином зображено на рисунку 2.3, а також детально пояснено його моменти.

Коли входить користувач як гість:

- Користувач має змогу переглянути товар який є в магазині;
- Користувач має змогу зареєструватись або увійти в систему;

Коли входить авторизований користувач:

- Клієнт має змогу переглянути товар який є в магазині;
- Клієнт має змогу переглядати особистий кабінет;
- Клієнт має доступ до корзини;
- Клієнт має змогу очистити або продовжити замовлення;
- Клієнт має змогу перегляд оформленого замовлення;

- Клієнт після отримання товару має змогу залишити відгук під товаром;

Коли входить адміністратор:

- Після авторизації адміністратором з'являється доступ до адмін панелі
- В панелі адміністратор може переглянути загальну статистику по інтернет магазину
- Адміністратор має змогу додавати товар, та окремо також видаляти та редагувати його
- У адміністратора також є доступ до перегляду замовлень
- Адміністратор має змогу редагувати статус доставки

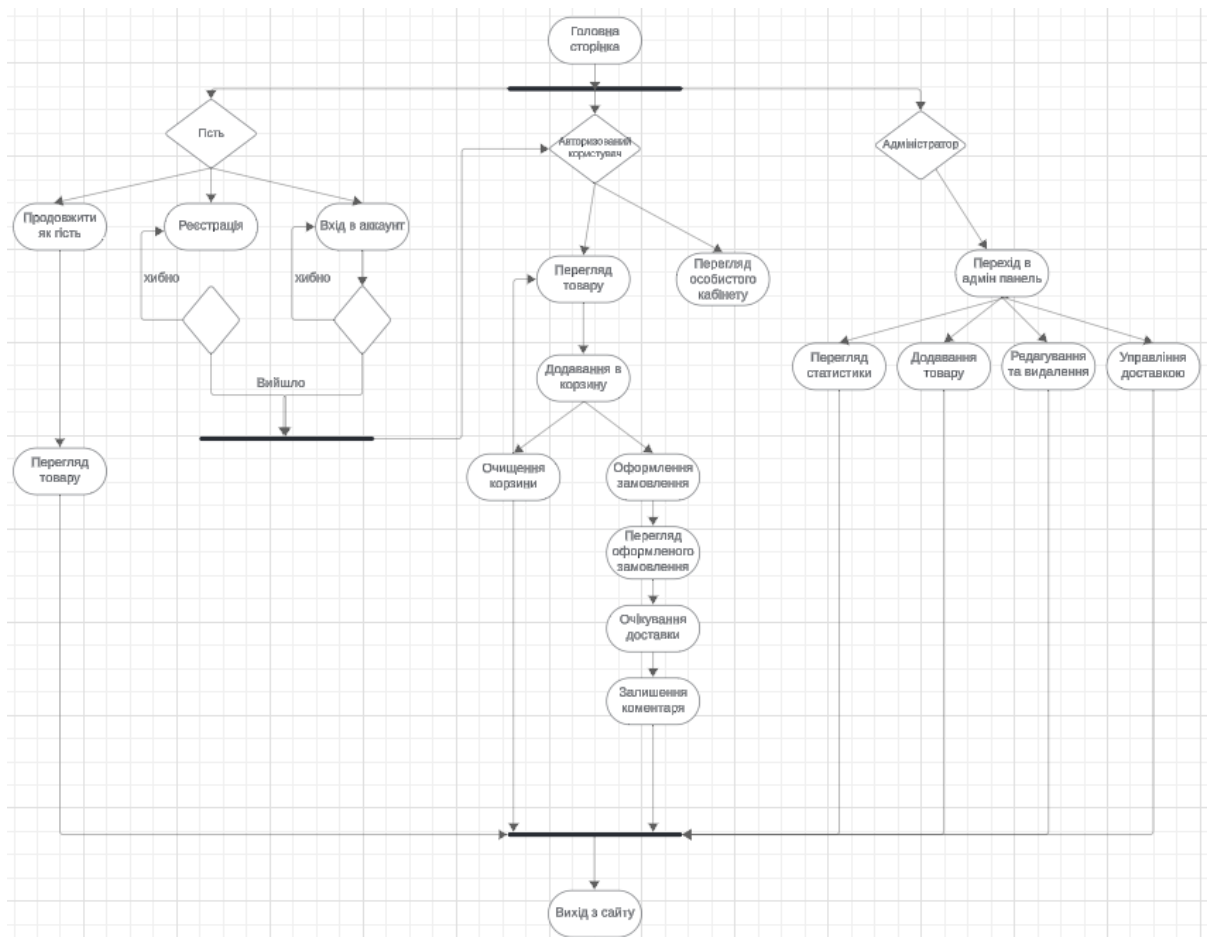


Рис 2.3 Діаграма дій користувачів на сайті

2.4 Реалізація функціоналу інтернет магазину з продажу ігор

Почнемо розглядати загальну роботу інтернет магазину. Все починається з входу на головну сторінку за якою йде перегляд товарів. Після чого потрібно увійти або зареєструватись в особистий кабінет. Який дасть змогу додавати товар до кошику. В кошику робити можливість перегляду інформації про купівлю та переходити до видалення списку товарів або до оформлення замовлення. В оформленні замовлення потрібно вказати всі необхідні данні для цього. Також після, авторизований користувач з роллю адміністратор зможе переглянути статус замовлення. Як інший функціонал додавати, редагувати, видаляти товар. Та загальну статистику сайту. Про це і не тільки розпишемо в цьому розділі.

Насамперед коли клієнт переходить на головну сторінку, було добавлено функцію різноманітного відображення товару на сайті, так званий рандом. Що кожного разу пропонуваний товар буде інший.

```
export default async function Home({searchParams}: HomeProps) {
  const products = await getProducts(searchParams)

  if(products.length === 0) {
    return <NullData title="Опа, Нема товару який ви шукали. Натисніть очищення фільтру"/>
  }

  function shuffleArray(array: any) {
    for(let i: number = array.length - 1; i>0; i--) {
      const j: number = Math.floor(Math.random() * (i+1));
      [array[i], array[j]] = [array[j], array[i]]
    }

    return array
  }

  const shuffledProducts = shuffleArray(products)

  return (<div className="p-8">
    <Container>
      <div>
        <HomeBanner/>
      </div>
      <div className="grid grid-cols-2 sm:grid-cols-3 lg:grid-cols-4 xl:grid-cols-5 2xl:grid-cols-6 gap-8">
        {shuffledProducts.map((product: ProductCardProps) =>
          {
            return <ProductCard data={product}/>;
          })}
      </div>
    </Container>
  </div>);
}
```

Рис 2.4 Функція рандома та загальне відображення товару на сторінці

Після чого потрібно зареєструватись або увійти на сайті в свій особистий кабінет. Для початку розглянемо реєстрацію. Цей код визначає функціональний компонент React під назвою RegisterForm, який використовує TypeScript для перевірки типів. Компонент приймає currentUser як пропс і обробляє реєстрацію користувача. Він використовує декілька React-хуків, включаючи useState для керування станом завантаження, useEffect для перенаправлення зареєстрованих користувачів та useForm з бібліотеки react-hook-form для обробки форми.

```
const RegisterForm: React.FC<RegisterFormProps> = ({currentUser}) => {

  const [isLoading, setIsLoading] = useState(false);
  const {register, handleSubmit, formState: {errors}} = useForm<FieldValues>({
    defaultValues:{
      name: '',
      email: '',
      password: ''
    }
  });

  const router = useRouter();

  useEffect (() => {
    if(currentUser){
      router.push("/cart");
      router.refresh();
    }
  },[]);

  const onSubmit:SubmitHandler<FieldValues> = (data) => {
    setIsLoading(true);
    axios.post ("/api/register", data).then(() => {
      toast.success("Аккаунт створено");

      signIn('credentials',{
        email: data.email,
        password: data.password,
        redirect: false,
      }).then((callback) => {
        if(callback?.ok){
          router.push("/cart");
          router.refresh();
          toast.success("Увійшли в");
        }

        if(callback?.error){
          toast.error(callback.error);
        }
      });
    }).catch(() => toast.error("Щось пішло не так")).finally( () => {
      setIsLoading(false);
    }
  );

  if(currentUser){
    return <p className="text-center">Перенаправлення після входу в систему</p>
  }
}
```

Рис 2.5 Функціонал реєстрації на сайті

Форма включає поля для введення імені, електронної пошти та пароля з валідацією та обробкою помилок. Коли форма надсилається, вона відправляє POST-запит на `/api/register` з даними форми, відображає повідомлення про успіх чи помилку за допомогою бібліотеки `toast` та намагається увійти в систему з використанням наданих електронної пошти та пароля. Якщо вхід успішний, користувача перенаправляють на сторінку `/cart`. Компонент також надає кнопку для реєстрації за допомогою Google за допомогою функції `signIn`.

Якщо `currentUser` вже визначений, компонент відображає повідомлення про перенаправлення після входу в систему. Форма відображає заголовок, кнопки, поля введення та посилання на сторінку входу.

```

if(currentUser){
  return <p className="text-center">Перенаправлення після входу в систему</p>
}

return (
  <>
    <Heading title="Реєстрація в Play VEXON"/>
    <Button
      outline
      label="Зареєструватись за допомогою Google"
      icon={AiOutlineGoogle}
      onClick={() => {
        signIn("google");
      }}
    />
    <hr className="bg-slate-300 w-full h-px"/>
    <Input
      id="name"
      label="Name"
      disabled={isLoading}
      register={register}
      errors={errors}
      required
    />
    <Input
      id="email"
      label="Email"
      disabled={isLoading}
      register={register}
      errors={errors}
      required
    />
    <Input
      id="password"
      label="Password"
      disabled={isLoading}
      register={register}
      errors={errors}
      required
      type="password"
    />
    <Button label = {isLoading ? "Завантаження" : "Sign Up"} onClick={handleSubmit(onSubmit)}/>
    <p className="text-sm">
      Маєте готовий аккаунт? <Link className="underline" href={'/login'}>
        Увійдіть
      </Link>
    </p>
  </> );
}

```

Рис 2.6 Загальна форма реєстрації

						ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
							34

Загальний вигляд входу має ідентичний вигляд реєстрації з похожим функціоналом. Тому перейдемо відразу до огляду коли клієнт натискає на товар який йому сподобався та переходить до його огляду. Тоді код для початку визначає початковий стан для cartProduct, що включає властивості продукту, такі як ідентифікатор, назва, опис, категорія, бренд, вибране зображення, кількість і ціна. Функція useEffect перевіряє, чи продукт вже є в кошику, і відповідно оновлює “isProductInCart”. Також обчислюється рейтинг продукту на основі відгуків. Функція “handleColorSelect” оновлює вибране зображення продукту.

```
const ProductDetails: React.FC<ProductDetailsProps> = ({ product }) => {
  const { handleAddProductToCart, cartProducts } = useCart();
  const [isProductInCart, setIsProductInCart] = useState(false);
  const [cartProduct, setCartProduct] = useState<CartProductType>({
    id: product.id,
    name: product.name,
    description: product.description,
    category: product.category,
    brand: product.brand,
    selectedImg: { ...product.images[0] },
    quantity: 1,
    price: product.price
  });

  const router = useRouter();

  useEffect(() => {
    setIsProductInCart(false);

    if (cartProducts) {
      const existingIndex = cartProducts.findIndex((item) => item.id === product.id);

      if (existingIndex > -1) {
        setIsProductInCart(true);
      }
    }
  }, [cartProducts]);

  const productRating = product.reviews.reduce((acc: number, item: any) => item.rating + acc, 0) / product.reviews.length;

  const handleColorSelect = useCallback((value: SelectedImgType) => {
    setCartProduct((prev) => {
      return { ...prev, selectedImg: value };
    });
  }, [cartProduct.selectedImg]);
};
```

Рис 2.7 Функціонал огляду товару

Також для зручності вибору кількості товару була введена функція “handleQtyDecrease” і “handleQtyIncrease”. Які в свою чергу дозволяють віднімати і додавати певну кількість товару для клієнта який він бажає. Головне обмеження це від 1 до 99.

```

const handleQtyIncrease = useCallback(() => {
  if (cartProduct.quantity === 99) {
    return;
  }

  setCartProduct((prev) => {
    return { ...prev, quantity: prev.quantity + 1 };
  });
}, [cartProduct]);

const handleQtyDecrease = useCallback(() => {
  if (cartProduct.quantity === 1) {
    return;
  }
  setCartProduct((prev) => {
    return { ...prev, quantity: prev.quantity - 1 };
  });
}, [cartProduct]);

```

Рис 2.8 Функціонал додавання та віднімання товару

Наступним що користувач буде робити це буде додавання товару в корзину та переходити до неї напряму. Тому перейдемо до огляду функціоналу корзини, а саме до головних компонентів. Розпочнемо з “handleAddProductToCart” , ця функція дозволяє додати в корзину товар з перевіркою чи існує товар чи ні і також зі зберіганням інформації в локальне сховище браузера під ключем “playShopCartItems”.

```

const handleAddProductToCart = useCallback((product: CartProductType) => {
  setCartProducts((prev) => {
    let updateCart;

    if (prev) {
      updateCart = [...prev, product];
    } else {
      updateCart = [product];
    }

    toast.success("Product added to cart")
    localStorage.setItem("playShopCartItems", JSON.stringify(updateCart))
    return updateCart;
  })
}, []);

```

Рис 2.9 Додавання товару в корзину

Якщо ж покупець не влаштує товар який він додав до кошика, то для цього є функція “handleRemoveProductFromCart”. На початку функція перевіряє чи існує “cartProducts” товар взагалі в корзині. Тільки після цього створює новий масив “filteredProducts” без товару який натиснули як видалити. Про успішне видалення буде сповіщати toast метод і оновлений масив зберігається в локальному сховищі браузера під ключем “playShopCartItems” оновлюючи попередні дані

```
const handleRemoveProductFromCart = useCallback((
  product: CartProductType) => {
    if(cartProducts){
      const filteredProducts = cartProducts.filter((item) => {
        return item.id !== product.id;
      })

      setCartProducts(filteredProducts);
      toast.success("Product removed");
      localStorage.setItem("playShopCartItems", JSON.stringify(filteredProducts));
    }
  }, [cartProducts])
```

Рис 2.10 Видалення товару з кошика

Також користувач матиме змогу видалити весь товар з кошика, якщо його не влаштує набір який він додав до нього. Таким чином він очистить сторінку та все що було в локальному сховищі, для запису нових даних.

```
const handleClearCart = useCallback(() => {
  setCartProducts(null)
  setCartTotalQty(0)
  localStorage.setItem("PlayShopCartItems", JSON.stringify(null));
}, [cartProducts])
```

Рис 2.11 Видалення всього товару з кошика

Наступне це опис буде оплати замовлення в якому при зміні товарів у кошику або платіжного наміру, виконується “useEffect”. Перевіряється наявність товарів у кошику. Якщо товари є, встановлюється стан завантаження і скидається стан помилки. Виконується POST-запит до сервера для створення платіжного наміру, передаючи товари з кошика і поточний платіжний намір. Обробляється відповідь: якщо статус відповіді 401, користувач

перенаправляється на сторінку входу, інакше відповідь перетворюється на JSON. Отримані дані використовуються для встановлення клієнтського секрету та платіжного наміру. У разі помилки встановлюється стан помилки, виводиться повідомлення в консоль і показується сповіщення користувачу. Створюється конфігурація для Stripe Elements з клієнтським секретом і темою оформлення "stripe" з плаваючими мітками. Використовується "useCallback" для визначення функції, яка встановлює успішний стан платежу на передане значення [21].

```
useEffect(() => {
  if(cartProducts){
    setLoading(true)
    setError(false)

    fetch("/api/create-payment-intent", {
      method: "POST",
      headers: {"Content-Type": "application/json"},
      body: JSON.stringify({
        items: cartProducts,
        payment_intent_id: paymentIntent,
      })
    }).then((res) => {
      setLoading(false);
      if(res.status === 401){
        return router.push("/login");
      }

      return res.json()
    }).then((data) => {
      setClientSecret(data.paymentIntent.client_secret);
      handleSetPaymentIntent(data.paymentIntent.id);
    }).catch((error) => {
      setError(true);
      console.log("Error", error);
      toast.error("Щось пішло не так");
    })
  }

}, [cartProducts, paymentIntent])

const options: StripeElementsOptions = {
  clientSecret,
  appearance: {
    theme: "stripe",
    labels: "floating",
  },
};

const handleSetPaymentSuccess = useCallback((value: boolean) => {
  setPaymentSuccess(value);
}, []);
```

Рис 2.12 Оплата замовлення

І як підсумок на прикінці в адмін панелі ми можемо використати функцію яка буде давати статистику по замовленням та товарам. Цей “useEffect” оновлює статистику при кожній зміні orders, products, або users. Він рахує загальну суму продажів (totalSale) і фільтрує замовлення на сплачені та несплачені. Потім оновлює значення з цими новими даними. Після оновлення створюється масив summaryKeys, який містить ключі об'єкта summaryData. І відображається як інформація на сайті

```
useEffect(()=>{
  setSummaryData ((prev) => {
    let tempData = {...prev}

    const totalSale = orders.reduce((acc,item) => {
      if(item.status === 'complete'){
        return acc + item.amount
      } else return acc
    }, 0)

    const paidOrders = orders.filter((order => {
      return order.status === 'complete'
    })))

    const unpaidOrders = orders.filter((order => {
      return order.status === 'в очікуванні'
    })))

    tempData.sale.digit = totalSale;
    tempData.orders.digit = orders.length;
    tempData.paidOrders.digit = paidOrders.length;
    tempData.unpaidOrders.digit = unpaidOrders.length;
    tempData.products.digit = products.length;
    tempData.users.digit = users.length;

    return tempData
  })
},[orders, products, users])

const summaryKeys = Object.keys(summaryData)
```

Рис 2.13 Збір статистики

Висновки до другого розділу

В розділі 2 зосереджувалися на проектуванні та розробці інтернет-магазину продажу ігор. Починаючи з визначення варіантів використання та об'єктно-орієнтованої структури, різні можливості та визначили оптимальні моделі для нашого магазину.

Детально було розроблено базу даних системи, визначивши необхідні таблиці, їхні взаємозв'язки та атрибути. Після цього перейшли до проектування та реалізації алгоритмів роботи інтернет-магазину. Розглянули ефективні алгоритми для операцій додавання товарів до кошика, оформлення замовлення, обробки платежів та управління інвентарем. Крім того, увага була на забезпеченні безпеки даних та операцій з платежами.

У заключному етапі реалізований функціонал інтернет-магазину з продажу ігор. Це включає в себе створення користувальницького інтерфейсу для перегляду товарів, додавання їх до кошика, оформлення замовлення, а також можливість аутентифікації користувачів та взаємодії з їхніми обліковими записами.

В результаті розділ 2 дипломної роботи допоміг глибше розібратися з проектуванням і розробкою інтернет-магазину, а також набути практичних навичок у сфері веб-розробки та управління базами даних.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ МАГАЗИНОМ ПРОДАЖУ ІГОР

3.1 Порядок встановлення та налаштування параметрів інтернет магазину

Конфігурація інтернет магазину з продажу ігор вимагає використання всіх її компонентів і налаштування як окремих елементів, так і всієї системи.

Щоб проілюструвати архітектуру інтернет магазину, було створено розгорнуто діаграму (рис 3.1.).

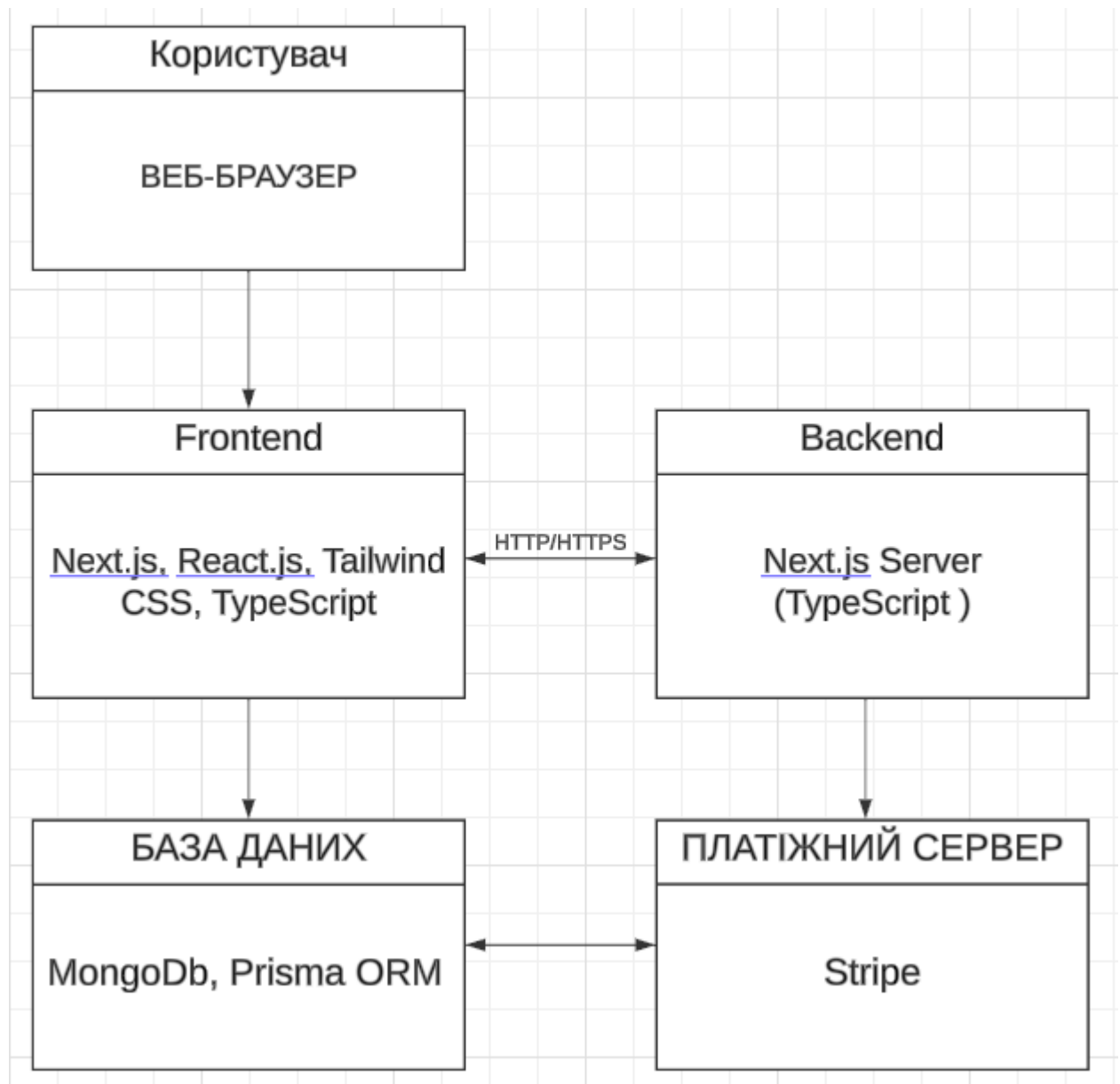


Рис 3.1 Діаграма розгортання інтернет магазину з продажу ігор

Опис компонентів:

- Користувач: Використовує веб-браузер для доступу до інтернет-магазину.
- Frontend:
 - Next.js: Фреймворк для React, що дозволяє серверний рендеринг та генерацію статичних сайтів.
 - React.js: Бібліотека для побудови користувацького інтерфейсу.
 - Tailwind CSS: Утилітарний CSS фреймворк для швидкої стилізації.
 - TypeScript: Надбудова над JavaScript для статичної типізації.
- Backend:
 - Next.js сервер: Використовується для обробки серверних запитів та рендерингу сторінок.
 - TypeScript: Використовується для статичної типізації.
- База даних:
 - MongoDB: Реляційна база даних.
 - Prisma ORM: Інструмент для роботи з базою даних [22 - 24].
- Платіжний сервер: Інтеграція зі Stripe API для обробки платежів.
- Хостинг платформи: Використання Vercel для хостингу Next.js додатку.

3.2 Структура інтерфейсу інтернет магазину продажу ігор

В цьому розділі розглянемо інтерфейс інтернет магазину з продажу ігор. Будуть продемонстровані різноманітні скріншоти та також індивідуальний опис до кожного з них. Зауважуючи кожен особливості та функціональну можливість розробленого веб застосунку. Рисунки які будуть наведені будуть демонструвати елементи користувацького інтерфейсу, з якими покупці будуть переглядати та взаємодіяти під час відвідування сайту.

Відкриття сайту: При заході на сайт, користувач бачить головну сторінку яка зустрічає його приємним візуальним інтерфейсом, який не давить на очі своєю кольоровою гамою (рис 3.2).

перевірити чи є вона по наявності в інтернет магазині або вона відсутня (рис 3.5 – 3.6).

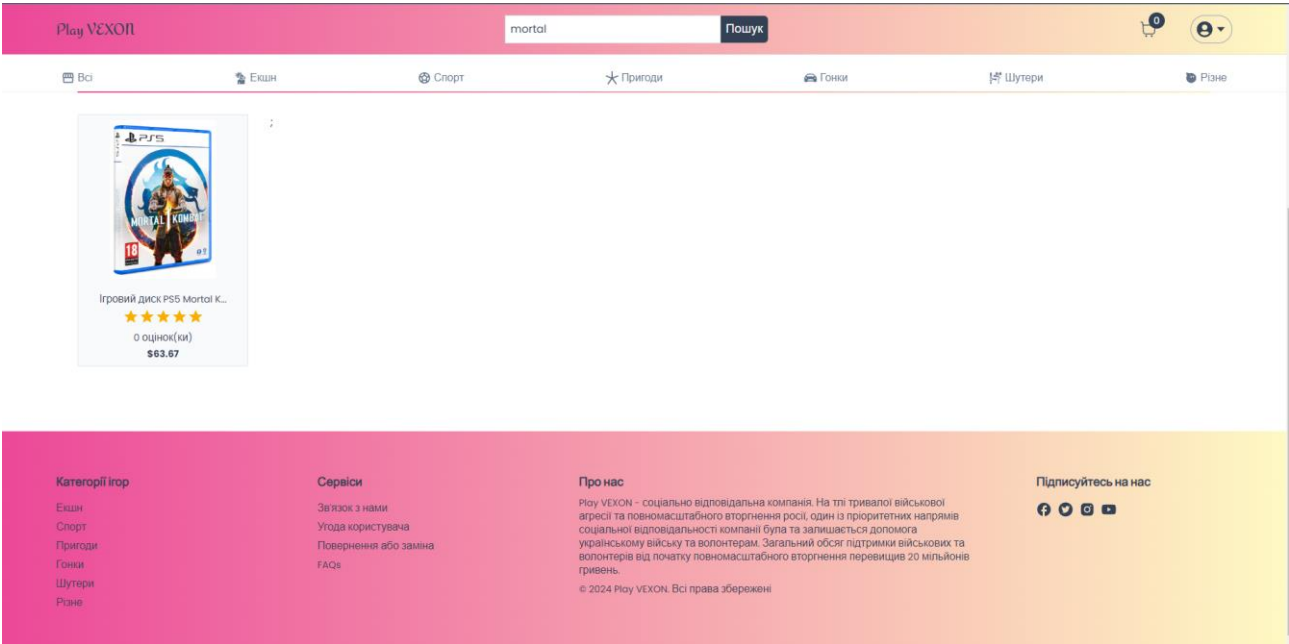


Рис 3.5 Результат пошуку гри яка є в наявності

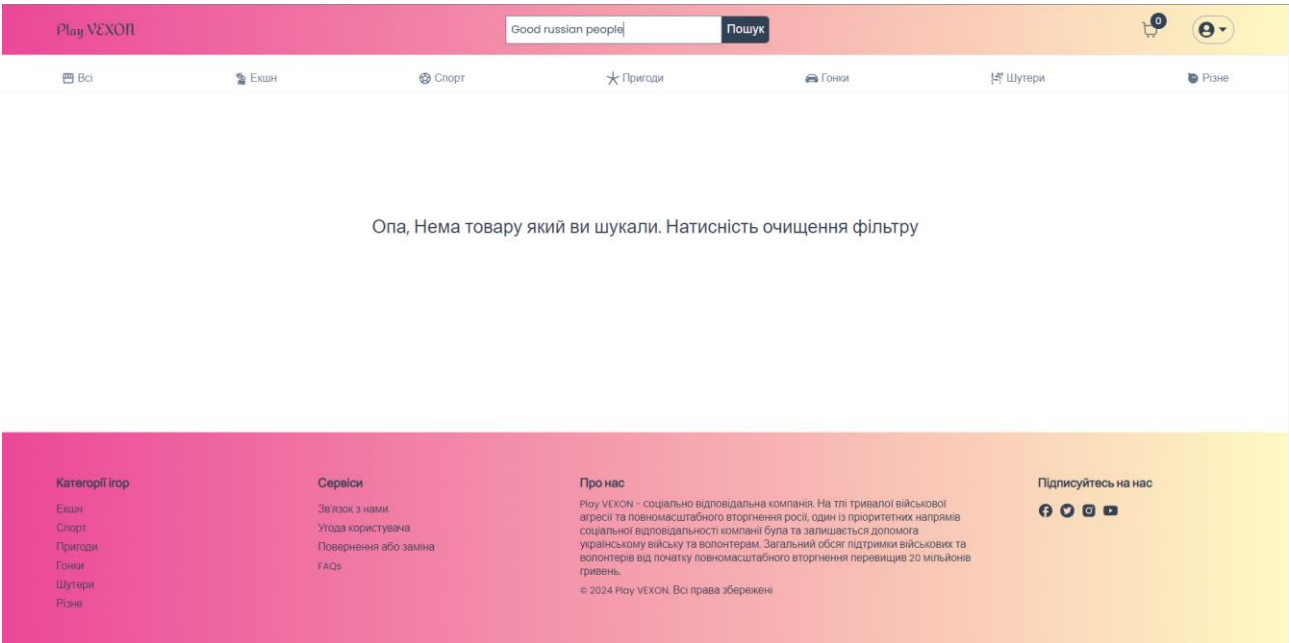


Рис 3.6 Результат пошуку гри так само як і людей не знайдено

Наступним кроком у покупця стане перехід на сторінку товару та можливість прочитати детально подробиці про гру яку він забажає придбати. Перевірити чи є вона в наявності та інші зображення самої гри, якщо вони є як додатково вказані (рис 3.7 – рис 3.8).

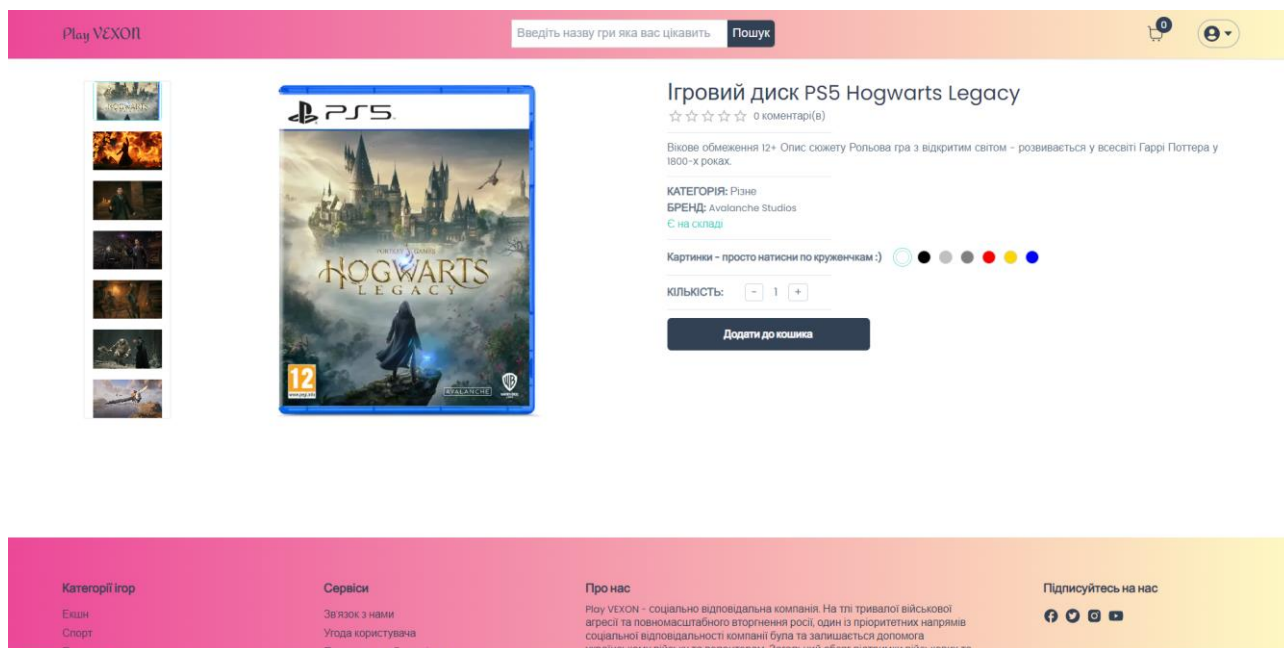


Рис 3.7 Огляд гри на персональній сторінці

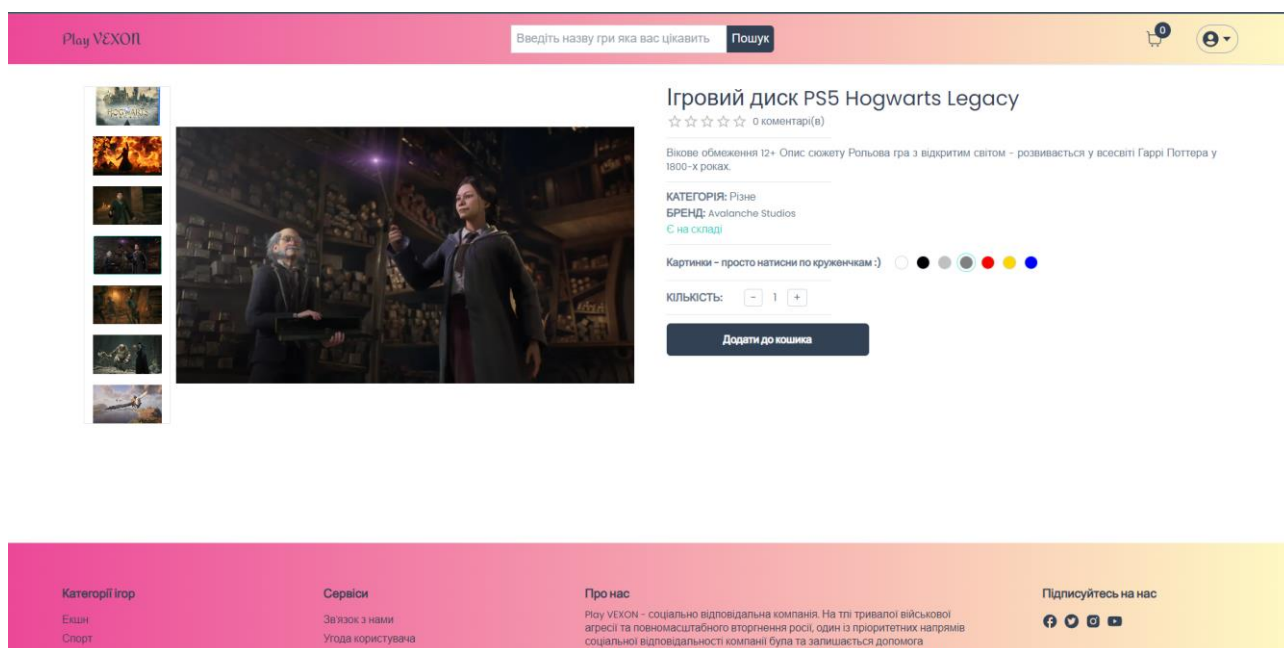


Рис 3.8 Огляд інших картинок

Надалі коли покупцю сподобалась гра, йому потрібно авторизуватись або зареєструватись на сайті. Для цього в правому верхньому кутку сайту є бічна панель яка допомагає це зробити (рис 3.9).

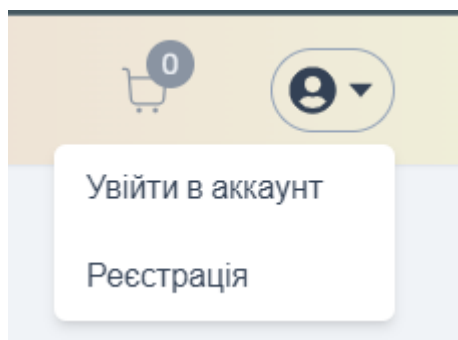


Рис 3.9 Меню входу в аккаунт

Коли користувач наприклад натискає “Увійти в аккаунт” то його перекидає на сторінку входу в аккаунт де він може по своїй електронній адресі та паролі авторизуватись та здійснити купівлю. Також є можливість авторизуватись через аккаунт “Google” якщо такий звісно є. А якщо новий користувач і він натиснув не на ту клавішу, то йому пропонується зареєструватись спочатку (рис 3.10 – рис 3.11).

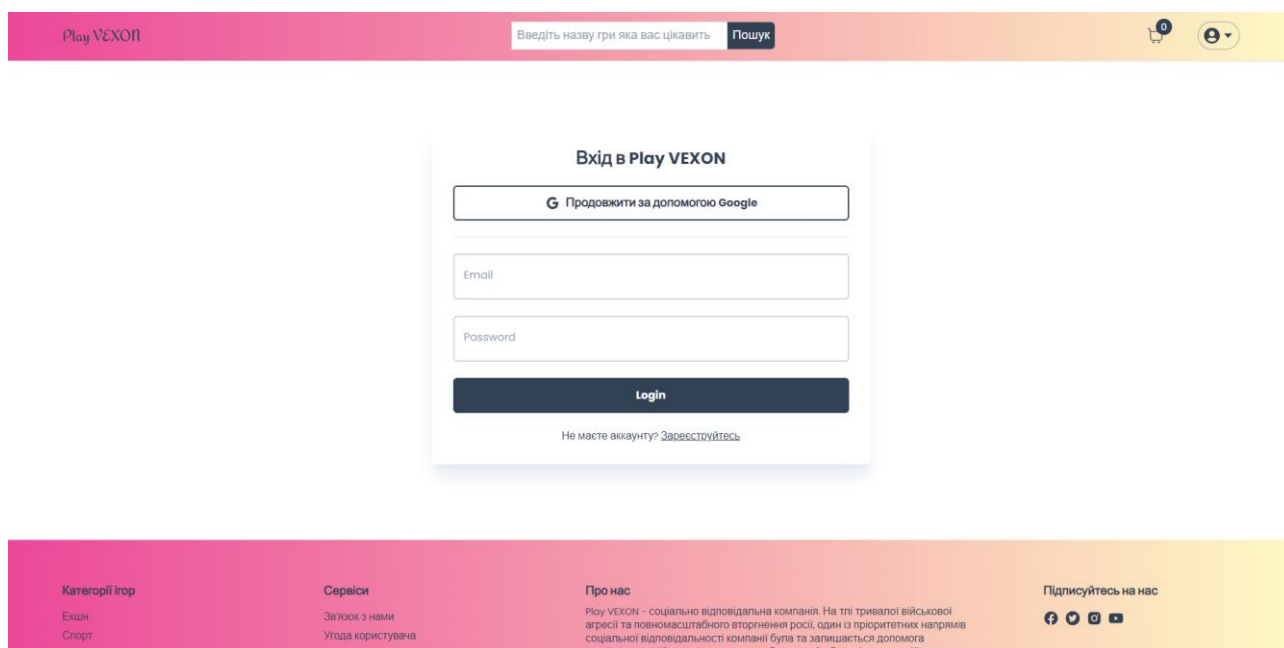


Рис 3.10 Вхід в особистий кабінет

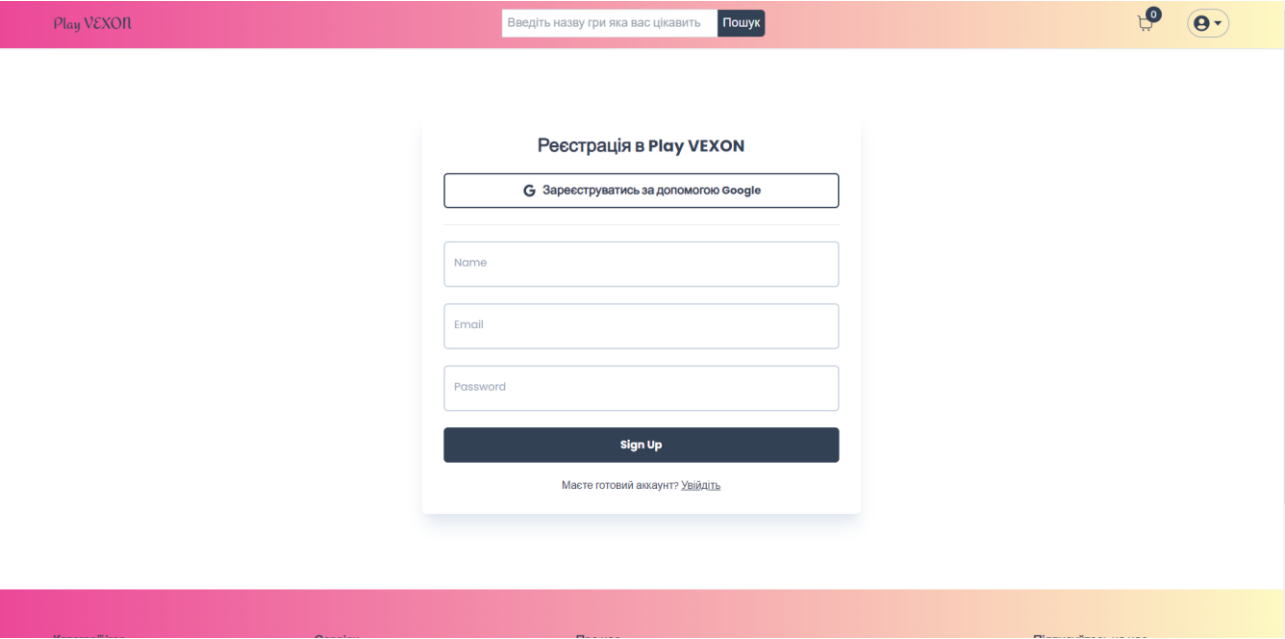


Рис 3.11 Реєстрація в інтернет магазині

Коли клієнт увійшов в свій профіль, його перекидає відразу в корзину, щоб він міг переглянути збережені товари які не купив або якщо немає товару, то йому пропонується повернутись на головну сторінку та спочатку додати товар який він бажає і тоді вже після цього повернутись до оформлення замовлення. Також в правому верхньому меню входу, з’являється фото користувача якщо він був авторизований через “Google” (рис 3.12).

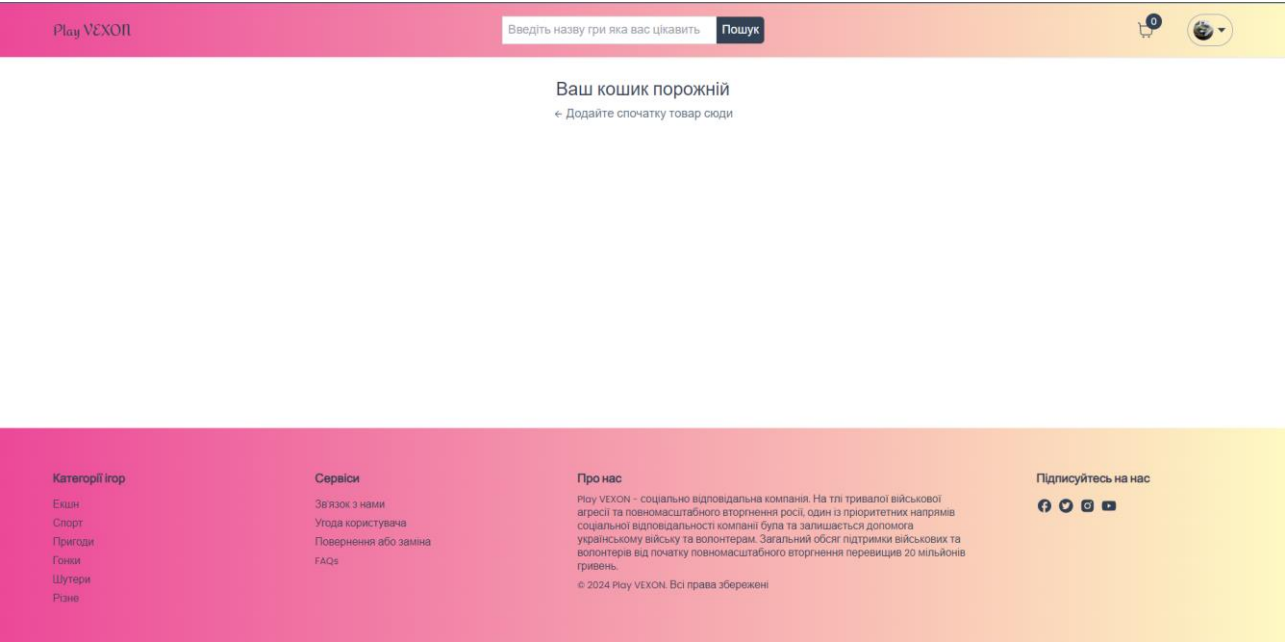


Рис 3.12 Перенаправлення після входу в акаунт

Коли покупець вибрав гру яку він хоче придбати, він вже переходить на сторінку корзини в якій є товар котрий був туди доданий. Де є можливість здійснити оформлення замовлення, очистити корзину та також змінити кількість ігор який клієнт бажає придбати. Або повернутись на головну сторінку щоб додати інші ігри які цікавлять (рис 3.13).

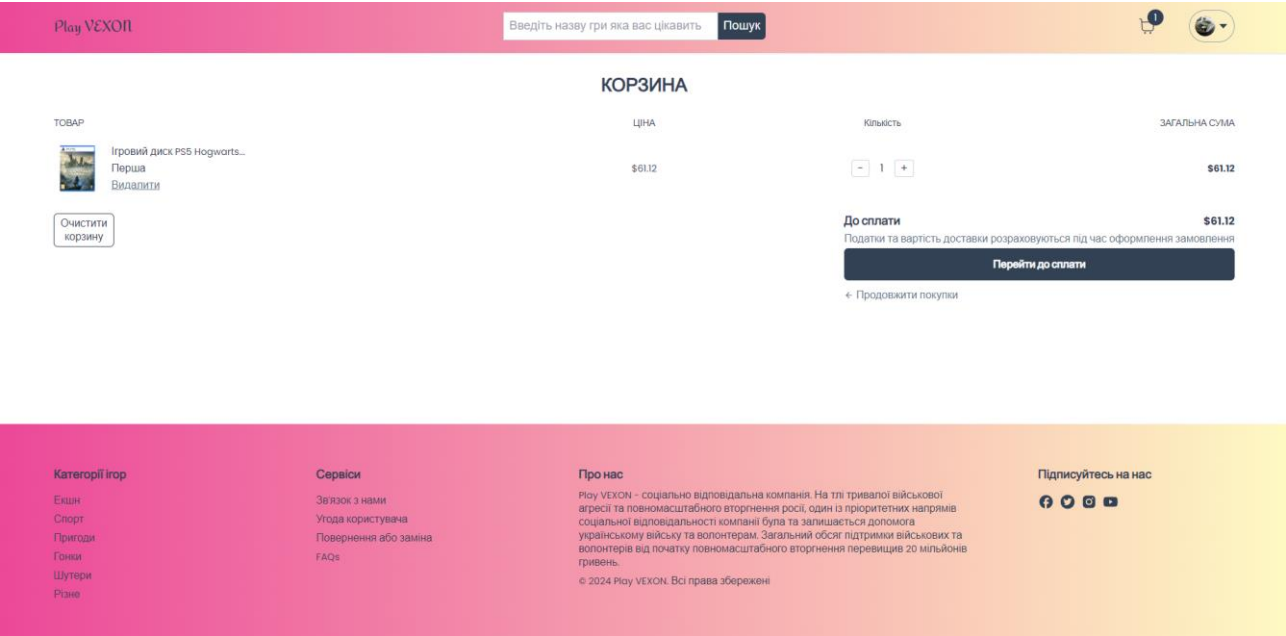


Рис 3.13 Корзина інтернет магазину

Якщо клієнта влаштовує товар який він обрав і нічого не бажає змінити, переходим до сторінки оформлення замовлення, де буде можливість вказати адресу доставку та також сплатити замовлення. Цей магазин орієнтований на багато країн, тому обов’язково потрібно буде вказати де проживаєш на даний момент (рис 3.14).

Play VEXON

Введіть назву гри яка вас цікавить **Пошук**

Інформація про адресу

Full name
Соколов Іван

Country or region
Ukraine

Address line 1
N/A

Address line 2
Вуля Вітрука 32

City
Житомир

Oblast
Житомирська область — Zhytomir

Postal code
10023

Інформація про оплату

Card number
4242 4242 4242 4242 **VISA**

Expiration
04 / 44

CVC
444

☒ Billing is same as shipping information

До сплати: \$61.12

Оплатити зараз

Рис 3.14 Заповнений рядок до сплати в оформленні замовлення

Після успішної оплати клієнту вибиває віконце про успішний платіж, після чого він може бути впевнений що отримає замовлення та перевірити статус своїх замовлень на сторінці “Перегляд ваших замовлень” (рис 3.15 – рис 3.16).

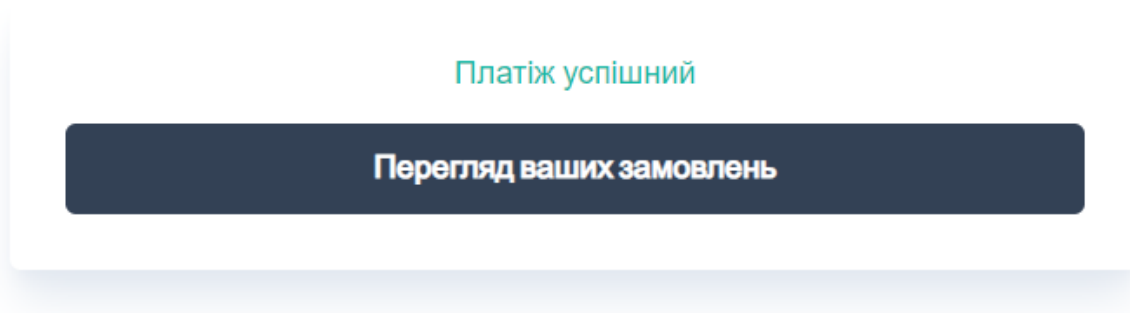


Рис 3.15 інформація про успішну оплату

Play VEXON

Введіть назву гри яка вас цікавить **Пошук**

Твої замовлення

☐	ID	Твій нікнейм	Ціна	Статус оплати	Статус доставки	Час замовлення	Інструменти
☐	6655b6a8218f93c3b92f4261	VEXON	\$61.00	сплачено ✓	в очікуванні	14 minutes ago	

Рис 3.16 Інформація в особистому кабінеті про статус замовлення

Також у покупця є можливість натиснути на клавiшу в вкладці “Інструменти” щоб переглянути більш детально інформацію про своєю замовлення (рис 3.17).

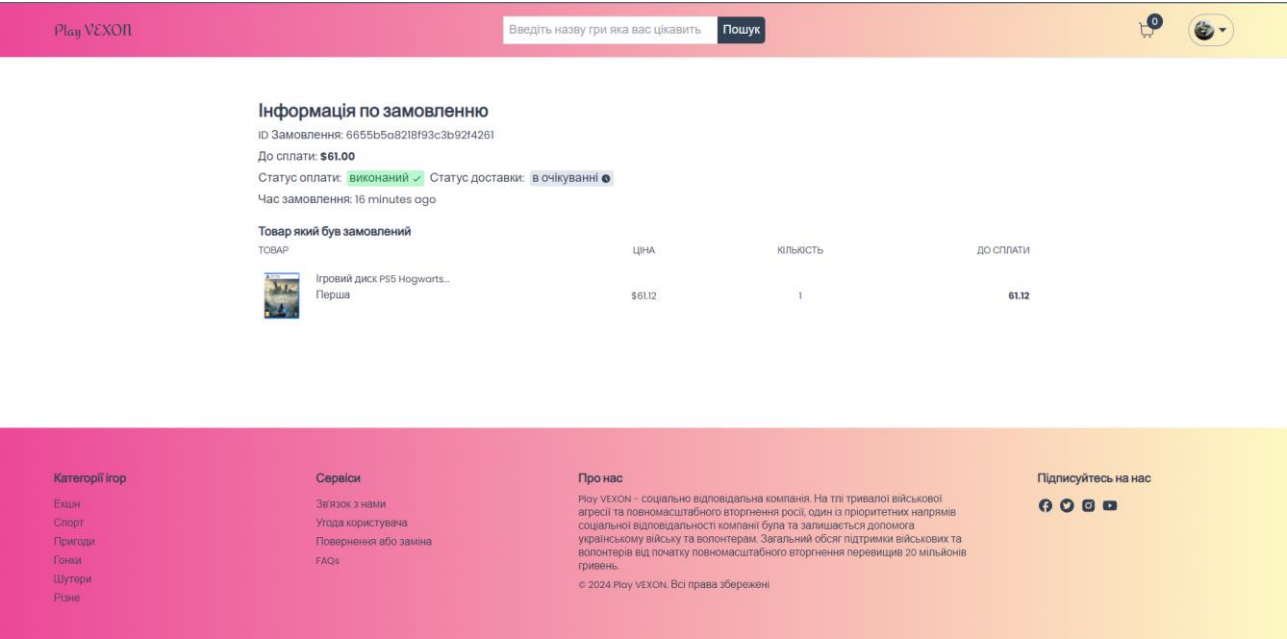


Рис 3.17 Інформація по замовленню

Далі так можна сказати “на сцену виходить” адміністратор. Це окремий авторизований користувач який має роль “ADMIN” та має доступ до товару, його редагуванню, додаванню та видаленню. Також по всій інформації та статистиці інтернет магазину. Розпочнемо з того що адміністратор в своєму меню користувача має окрему вкладку “Панель адміністратора” (рис 3.18.).

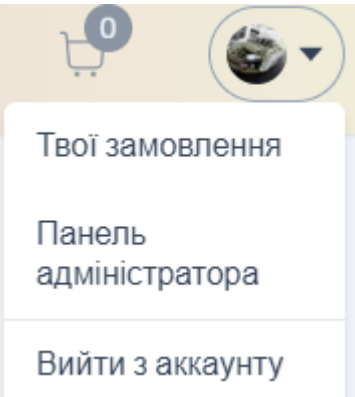


Рис 3.18 Меню авторизованого адміністратора

Перейшовши в адміністративну панель, перше на що ми натикаємося це статистика інтернет магазину, в якій відображається кількість товару який продали за весь час існування, скільки унікальних є товарів на сайті. Всього замовлень, а також скільки з них було успішно сплачених та не оплачених. І саме головне дійсних користувачів на сайті (рис 3.19).

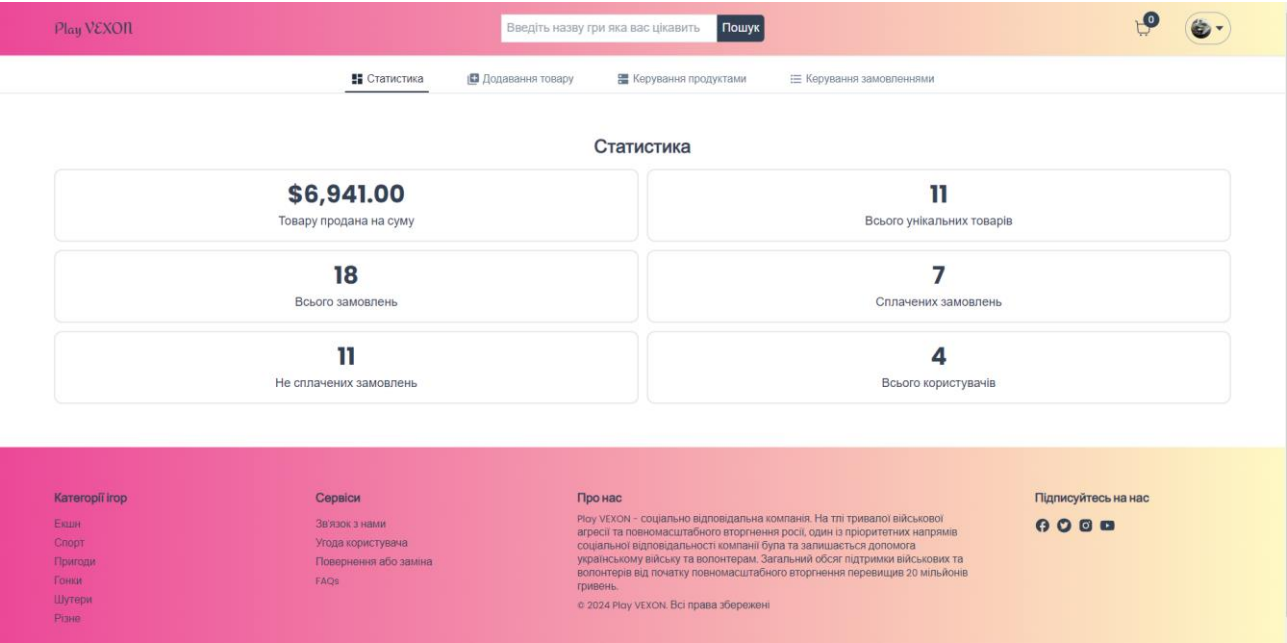


Рис 3.19 Сторінка статистики в адміністративній панелі

Далі адміністратор може перейти на вкладку в меню “Додавання товару” де може вказати всю головну інформацію про гру. Конкретніше назву, ціну, бренд, опис, наявність, категорію, зображення та сама кнопка додавання товару (рис 3.20 – рис 3.21).

Рис 3.20 Перша частини додавання товару

Рис 3.21 Друга частина додавання товару

Наступним етапом після додавання товаром, якщо товар вже не потрібний на сайті то видалити його, закінчився по наявності або потрібно його відредагувати. Адміністратор може звернутись до сторінки “Керування продуктами” на якій зможе переглянути дійсний товар на сайті та вибрати операцію яку потрібно зробити з товаром через панель інструментів (рис 3.22). Також як приклад буде відображена зміна активності по наявності товару та пуш сповіщення на сайті, через інструмент зображений приклад на рисунку 3.23.

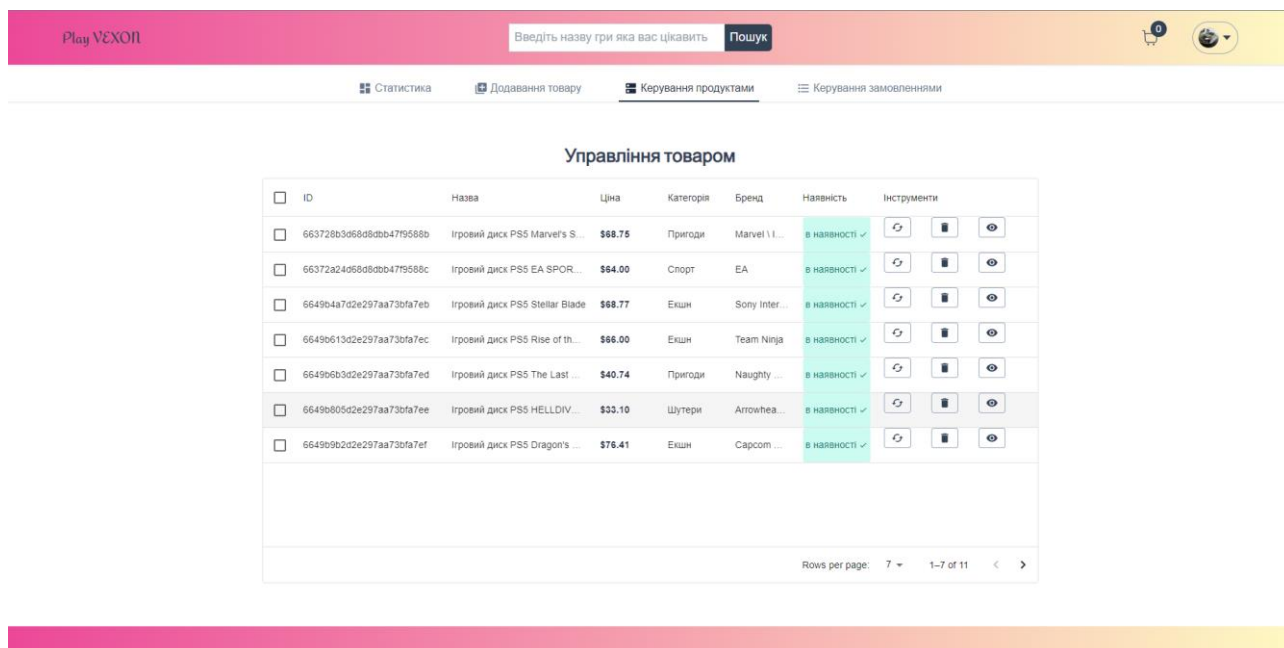


Рис 3.22 Сторінка управління товаром

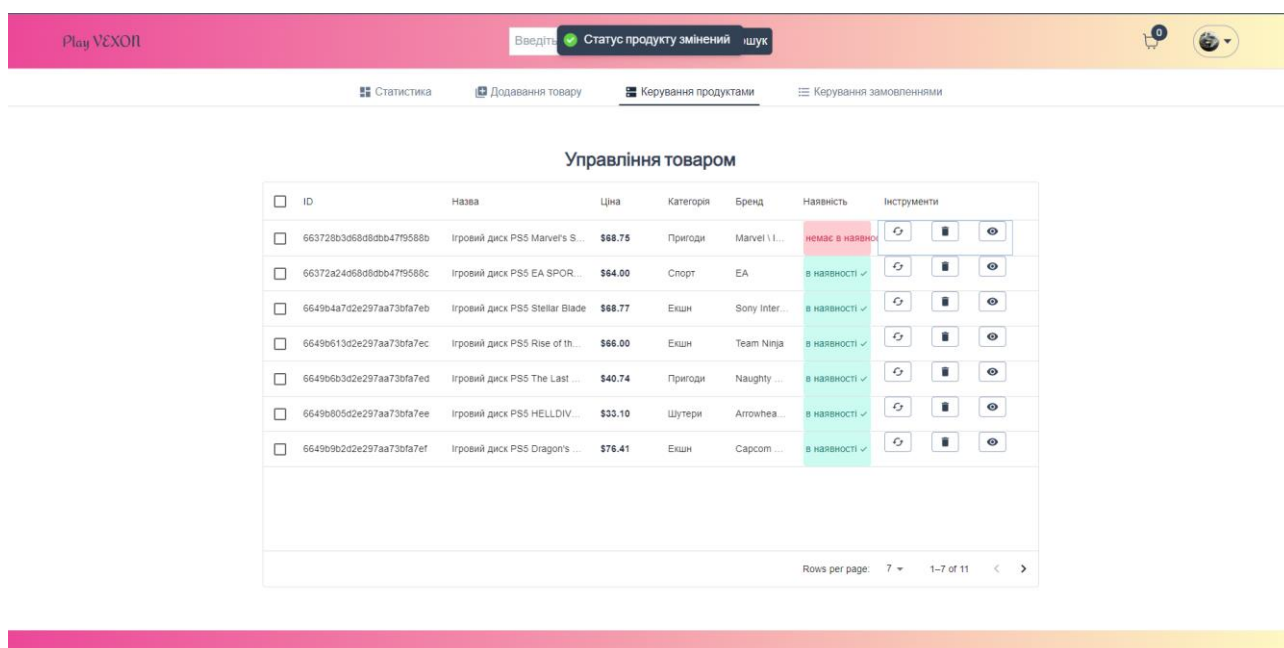


Рис 3.23 Приклад зміни наявності товару в асортименті

Також відразу через інструмент можна відразу перейти на сторінку гри, та перевірити чи змінилась для користувача відображення інформації (рис 3.24).



Ігровий диск PS5 Marvel's Spider-Man 2

☆☆☆☆☆ 0 коментарі(в)

Вікове обмеження:16+ Опис сюжету: Двоє протагоністів гри, Пітер Паркер і Майлз Моралес, мають битися з суперлиходіями Крейвен-мисливцем і Ящером, а також зіткнутися з Веномом. Дата релізу в Україні: 2023
Дата релізу в світі: 2023

КАТЕГОРІЯ: Пригоди

БРЕНД: Marvel \ Insomniac Games

Немає на складі

Картинки - просто натисни по кружечкам :)



КІЛЬКІСТЬ: - 1 +

Рис 3.24 Зміна інформації на сторінці товару

І остання вкладка вибору яку має адміністратор це “Керування замовленнями” на якій відображається інформацію про те, хто замовляв, що замовляли, статус оплати (рис 3.25). А також по інформації яка поступає від складу та кур’єрської доставки змінювати статус замовлення на відправлене та чи воно доставлене (рис 3.26 – рис 3.27).

Play VEXON

Введіть назву гри яка вас цікавить

Пошук

0

Статистика

Додавання товару

Керування продуктами

Керування замовленнями

Керування замовленнями

☐	ID	Ім'я клієнта	Сума	Статус платежу	Статус доставки	Дата	Інструменти		
☐	665505a8218f93c3b92f4261	VEXON	\$61.00	completed ✓	в очікуванні	an hour ago			
☐	665505a8218f93c3b92f4260	VEXON	\$61.00	completed ✓	в очікуванні	an hour ago			
☐	6637239a95d8dbb47f95889	VEXON	\$127.00	в очікуванні	в очікуванні	23 days ago			
☐	6637239a95d8dbb47f9588a	VEXON	\$61.00	в очікуванні	в очікуванні	23 days ago			
☐	662f5a866a470623248f43a8	VEXON	\$512.00	completed ✓	доставлено ✓	a month ago			
☐	662f5a866a470623248f43a7	VEXON	\$512.00	в очікуванні	в очікуванні	a month ago			
☐	662f59fb8a470623248f43a6	VEXON	\$127.00	в очікуванні	в очікуванні	a month ago			
Rows per page: 7 1-7 of 18									

Рис 3.25 Сторінка керування замовленнями

Play VEXON

Введіть Замовлення відправлено шук 0

Статистика

Додавання товару

Керування продуктами

Керування замовленнями

Керування замовленнями

☐	ID	Ім'я клієнта	Сума	Статус платежу	Статус доставки	Дата	Інструменти
☐	6655b5a8218f93c3b92f4261	VEXON	\$61.00	completed ✓	відправлено 41	an hour ago	+ ✓
☐	6655b5a8218f93c3b92f4260	VEXON	\$61.00	completed ✓	в очікуванні 41	an hour ago	+ ✓
☐	6637239ad58d8db47f95889	VEXON	\$127.00	в очікуванні 41	в очікуванні 41	23 days ago	+ ✓
☐	6637239ad58d8db47f9588a	VEXON	\$61.00	в очікуванні 41	в очікуванні 41	23 days ago	+ ✓
☐	662f5a868a470623248f43a8	VEXON	\$512.00	completed ✓	доставлено ✓	a month ago	+ ✓
☐	662f5a868a470623248f43a7	VEXON	\$512.00	в очікуванні 41	в очікуванні 41	a month ago	+ ✓
☐	662f59fb8a470623248f43a6	VEXON	\$127.00	в очікуванні 41	в очікуванні 41	a month ago	+ ✓

Рис 3.26 Зміна статусу доставки на відправлене

Play VEXON

Введіть

Замовлення доставлено

шук

0

Статистика

Додавання товару

Керування продуктами

Керування замовленнями

Керування замовленнями

☐	ID	Ім'я клієнта	Сума	Статус платежу	Статус доставки	Дата	Інструменти
☐	6655b5a8218f93c3b92f4261	VEXON	\$61.00	completed ✓	доставлено ✓	an hour ago	41 ✓ 🔍
☐	6655b5a8218f93c3b92f4260	VEXON	\$61.00	completed ✓	в очікуванні 1	an hour ago	41 ✓ 🔍
☐	6637239ad58d8db47f95889	VEXON	\$127.00	в очікуванні 1	в очікуванні 1	23 days ago	41 ✓ 🔍
☐	6637239ad58d8db47f9588a	VEXON	\$61.00	в очікуванні 1	в очікуванні 1	23 days ago	41 ✓ 🔍
☐	662f5a868a470623248f43a8	VEXON	\$512.00	completed ✓	доставлено ✓	a month ago	41 ✓ 🔍
☐	662f5a868a470623248f43a7	VEXON	\$512.00	в очікуванні 1	в очікуванні 1	a month ago	41 ✓ 🔍
☐	662f59fb8a470623248f43a6	VEXON	\$127.00	в очікуванні 1	в очікуванні 1	a month ago	41 ✓ 🔍
Rows per page: 7 1-7 of 18 < >							

Рис 3.27 Зміна статусу доставки на доставлене

3.3 Тестування роботи інтернет магазину з продажу ігор

Тестування роботи інтернет-магазину з продажу ігор включає різні аспекти функціональності, безпеки, зручності використання та сумісності.

В першу чергу було проведено функціональне тестування спрямоване на перевірку відповідності системи заявленим вимогам і коректної роботи всіх функцій. Це включає тестування реєстрації та входу, де перевіряється реєстрація нового користувача з різними типами інформації, вхід користувача з правильними та неправильними обліковими даними. Каталог ігор тестується на

можливість перегляду списку ігор, фільтрації, пошуку за назвою, а також детальну сторінку гри, де перевіряються опис, скріншоти, відео та відгуки. У кошику було протестовано додавання та видалення ігор, зміна кількості одиниць та перевірка загальної вартості замовлення. Оформлення замовлення перевірялось на заповнення та валідацію форми замовлення, онлайн оплати, підтвердження замовлення та отримання сповіщення про вдалу купівлю. Обліковий запис користувача перевірявся на можливість перегляду профілю та перегляду історії замовлень.

Наступним кроком перевіряється безпека, було перевірено захищеність системи від потенційних загроз та вразливостей. Включало перевірку процесу аутентифікації та авторизації, зокрема захищеність процесу входу та реєстрації. Захист даних включав перевірку шифрування особистих даних користувачів та безпечного зберігання паролів. Тестування на вразливості охоплював перевірки на XSS (міжсайтові скрипти) та CSRF (міжсайтові запити).

Далі була здійснена перевірка зручності використання інтерфейсу для кінцевих користувачів. Це включало оцінку легкості навігації по сайту, логічності та інтуїтивності меню, зручності процесу оформлення замовлення, зокрема кількості кроків для його завершення та зручності форми заповнення. Також оцінювалось візуальна привабливість сайту та його відповідність загальним стандартам і очікуванням користувачів.

Після цього перевірялась сумісність для забезпечення коректної роботи сайту на різних пристроях, браузерях та операційних системах. Це включало тестування на різних браузерях (Chrome, Firefox, Safari, Edge) і їх мобільних версіях, на різних пристроях (ПК, ноутбуки, планшети, смартфони) та операційних системах (Windows, macOS, Linux, iOS, Android).

Тестування інтеграцій перевіряло правильність роботи з зовнішніми системами та сервісами. Це включає перевірку інтеграції з платіжними системами (PayPal, кредитні картки, банківські перекази) та коректність обробки транзакцій. Також перевіряється авторизація через соціальні мережі (Google) і можливість ділитися контентом в соціальних мережах.

І на останок було проведено регресійне тестування яке мало на меті переконатися, що нові зміни в коді не вплинуть на раніше працюючий функціонал. Це включало впровадження автоматизованих тестів для ключових сценаріїв та їх регулярне виконання при внесенні змін.

Висновки до розділу 3

Розділ 3 розглядає ключові аспекти, пов'язані з інтерфейсом та порядком роботи інтернет-магазину з продажу ігор, а також тестування його функціональності.

У підрозділі 3.1 описано порядок встановлення та налаштування параметрів інтернет-магазину. Встановлення включає базову конфігурацію системи, налаштування платіжних модулів, інтеграцію з базами даних та налаштування захисних механізмів. Важливими аспектами є забезпечення безпеки транзакцій та збереження особистих даних користувачів.

У підрозділі 3.2 викладено структуру інтерфейсу інтернет-магазину. Інтерфейс повинен бути інтуїтивно зрозумілим, зручним у використанні та забезпечувати простоту навігації. Описано основні компоненти інтерфейсу, включаючи головну сторінку, каталог ігор, кошик, сторінку оформлення замовлення, особистий кабінет користувача та адміністративну панель для управління магазином.

Підрозділ 3.3 присвячено тестуванню роботи інтернет-магазину. Тестування включає функціональні перевірки, оцінку зручності використання, сумісності з різними пристроями та браузерами, а також перевірку безпеки. Важливим етапом є регресійне тестування, яке забезпечує стабільність роботи системи після внесення змін.

Таким чином, у розділі 3 описано комплексний підхід до створення, налаштування та тестування інтернет-магазину з продажу ігор. Дотримання всіх вказаних процедур дозволяє забезпечити ефективну, безпечну та зручну для користувачів роботу магазину, що є ключовим фактором для успішного функціонування бізнесу в сфері електронної комерції.

ВИСНОВКИ

Ця випускна кваліфікаційна робота присвячена створенню, аналізу та тестуванню інтернет-магазину з продажу ігор. В рамках дослідження були розглянуті різні аспекти проектування та реалізації такого магазину, починаючи з аналізу напрямків застосування, закінчуючи тестуванням функціоналу та зручності використання. Робота структурована на три основні розділи, кожен з яких має свої завдання та результати.

У першому розділі була поставлена задача розробки інтернет-магазину з продажу ігор, проведено аналіз аналогічних програмних продуктів та обґрунтовано вибір інструментальних засобів і вимог до апаратного забезпечення. Проведений аналіз дозволив виявити сильні та слабкі сторони існуючих рішень на ринку, а також визначити ключові вимоги до функціоналу і дизайну майбутнього продукту. На основі цього було обрано оптимальні технології та платформи для розробки інтернет-магазину, що забезпечують високу продуктивність та безпеку системи.

Другий розділ присвячений проектуванню інтернет-магазину. Було визначено варіанти використання та об'єктно-орієнтовану структуру системи, розроблено базу даних та алгоритми роботи магазину. Розробка бази даних включала визначення основних сутностей та зв'язків між ними, що забезпечує ефективне управління інформацією про товари, користувачів та замовлення. У процесі проектування алгоритмів роботи магазину була реалізована логіка обробки замовлень, управління кошиком, обробки платежів та інших ключових функцій системи. Важливою частиною цього розділу стала реалізація функціоналу магазину, який включає інтерфейс користувача, систему реєстрації та авторизації, а також адміністративну панель для управління контентом.

Третій розділ охоплює аспекти інтерфейсу та порядку роботи з інтернет-магазином, включаючи встановлення та налаштування параметрів системи, структуру інтерфейсу та тестування роботи магазину. Встановлення та налаштування параметрів магазину передбачає початкову конфігурацію

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						58

системи, налаштування платіжних модулів та захисних механізмів. Описано структуру інтерфейсу, яка повинна бути інтуїтивно зрозумілою та зручною для користувачів, забезпечуючи легкість навігації та доступ до всіх необхідних функцій. Тестування роботи магазину включає функціональні перевірки, оцінку зручності використання, сумісності з різними пристроями та браузерами, а також перевірку безпеки. Ретельне тестування дозволило виявити та усунути можливі проблеми, забезпечуючи стабільну та безпечну роботу системи.

Проведене дослідження показало, що успішна реалізація інтернет-магазину з продажу ігор вимагає комплексного підходу, який включає аналіз ринку, ретельне проектування системи, вибір оптимальних інструментів та технологій, а також повномасштабне тестування. Розроблений інтернет-магазин відповідає сучасним вимогам до функціональності, безпеки та зручності використання, що робить його конкурентоспроможним на ринку електронної комерції.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React SDK - [Електронний ресурс] - Режим доступу до ресурсу:
<https://portal.thirdweb.com/typescript/v5/react>
2. React hot toast - [Електронний ресурс] - Режим доступу до ресурсу:
<https://react-hot-toast.com/>
3. KeyBox - [Електронний ресурс] - Режим доступу до ресурсу:
<https://keybox.com.ua/>
4. Mgames - [Електронний ресурс] - Режим доступу до ресурсу:
<https://mgames.ua/>
5. GameShop - [Електронний ресурс] - Режим доступу до ресурсу:
<https://game-shop.com.ua/ua>
6. Retromagaz - [Електронний ресурс] - Режим доступу до ресурсу:
<https://retromagaz.com/>
7. I-gamer - [Електронний ресурс] - Режим доступу до ресурсу:
<https://i-gamer.ua/>
8. NextJS - [Електронний ресурс] - Режим доступу до ресурсу:
<https://nextjs.org/>
9. NextJS SSR - [Електронний ресурс] - Режим доступу до ресурсу:
<https://nextjs.org/docs/pages/building-your-application/rendering/server-side-rendering>
10. NextJS Routing - [Електронний ресурс] - Режим доступу до ресурсу:
<https://nextjs.org/docs/app/building-your-application/routing>
11. NextJS Rendering - [Електронний ресурс] - Режим доступу до ресурсу:
<https://nextjs.org/docs/app/building-your-application/rendering>
12. NextJs Optimizations - [Електронний ресурс] - Режим доступу до ресурсу:
<https://nextjs.org/docs/app/building-your-application/optimizing>
13. MongoDB - [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.mongodb.com/>
14. MongoDB Atlas - [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.mongodb.com/docs/atlas/>

15. TypeScript - [Электронный ресурс] - Режим доступа до ресурсу:
<https://www.typescriptlang.org/>
16. TypeScript for JavaScript - [Электронный ресурс] - Режим доступа до
ресурсу: <https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>
17. TypeScript Classes - [Электронный ресурс] - Режим доступа до ресурсу:
<https://www.typescriptlang.org/docs/handbook/2/classes.html>
18. TailwindCSS - [Электронный ресурс] - Режим доступа до ресурсу:
<https://tailwindcss.com/docs/guides/nextjs>
19. Google Cloud - [Электронный ресурс] - Режим доступа до ресурсу:
<https://console.cloud.google.com/welcome?hl=ru&project=play-shoper>
20. NodeJS – [Электронный ресурс] - Режим доступа до ресурсу:
<https://nodejs.org/en/learn/getting-started/an-introduction-to-the-npm-package-manager>
21. Stripe - [Электронный ресурс] - Режим доступа до ресурсу:
<https://dashboard.stripe.com/>
22. Prisma – [Электронный ресурс] - Режим доступа до ресурсу:
<https://www.prisma.io/>
23. Prisma ORM - [Электронный ресурс] - Режим доступа до ресурсу:
https://www.prisma.io/orm?utm_source=website&utm_medium=banner&utm_id=pulse-dg
24. Learn Prisma – [Электронный ресурс] - Режим доступа до ресурсу:
https://www.prisma.io/learn?utm_source=website&utm_medium=banner&utm_id=pulse-dg

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						62

Реалізація сторінки яка відповідає за технічну сторону головної сторінки.

Лістинг layout.tsx

```
import type { Metadata } from "next";
import { Poppins } from "next/font/google";

import "./globals.css";

import NavBar from "./components/nav/NavBar";
import Footer from "./components/footer/Footer";
import CartProvider from "@providers/CartProvider";
import { Toaster } from "react-hot-toast";

const poppins = Poppins({ subsets: ["latin"], weight: ['400', '700'] });

export const metadata: Metadata = {
  title: "Play VEXON",
  description: "Play app",
};

export default async function RootLayout({
  children,
}): {
  children: React.ReactNode;
}) {
  return (
    <html lang="en">
      <body className={` ${poppins.className} text-slate-700`}>
        <Toaster toastOptions={{
          style:{
            background: "rgb(51 65 85)",
            color: "#fff",
          }
        }}/>
```

```

    <CartProvider>

    <div className="flex flex-col min-h-screen">

      <NavBar/>

      <main className="flex-grow">{children}</main>

      <Footer/>

    </div>

  </CartProvider>

</body>

</html>

);
}

```

Додаток Б

Реалізація функції, що відповідає за замовлення клієнта

Лістинг OrderDetails.tsx

```

"use client";

import Heading from "@app/components/Heading";
import Status from "@app/components/Status";
import { formatPrice } from "@utils/formatPrice";
import { Order } from "@prisma/client";
import moment from "moment";
import React from "react";
import { MdAccessTimeFilled, MdDeliveryDining, MdDone } from "react-icons/md";
import OrderItem from "../OrderItem";

interface OrderDetailsProps{
  order: Order;
}

const OrderDetails: React.FC<OrderDetailsProps> = ({order}) => {

  return ( <div className="max-w-[1150px] m-auto flex flex-col gap-2">

    <div className="mt-8">

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						64

```

        <Heading title="Інформація по замовленню"/>

    </div>

    <div>ID Замовлення: {order.id}</div>

    <div>До сплати: {" "} <span className="font-bold">{formatPrice(order.amount /
100)}</span></div>

    <div className="flex gap-2 items-center">

        <div>Статус оплати:</div>

        <div>

            {order.status === "в очікуванні"? <Status text="в очікуванні"
            icon={MdAccessTimeFilled} bg="bg-slate-200" color="text-slate-700"/>: order.status === "complete"
            ?

                <Status text="виконаний" icon={MdDone} bg="bg-green-200" color="text-green-700"/>
            : <></> }

        </div>

        <div>Статус доставки:</div>

        <div>

            {order.deliveryStatus === "в очікуванні"? <Status text="в очікуванні"
            icon={MdAccessTimeFilled} bg="bg-slate-200" color="text-slate-700"/>: order.deliveryStatus ===
            "відправлено" ?

                <Status text="відправлено" icon={MdDeliveryDining} bg="bg-purple-200"
            color="text-purple-700"/> : order.deliveryStatus === "доставлено" ?

                <Status text="доставлено" icon={MdDone} bg="bg-green-200" color="text-green-
            700"/> : <></> }

        </div>

    </div>

    <div>Час замовлення: {moment(order.createdAt).fromNow()}</div>

    <div>

        <h2 className="font-semibold mt-4 mb-2">Товар який був замовлений</h2>

        <div className="grid grid-cols-5 text-xs gap-4 pb-2 itemc-center">

            <div className="col-span-2 justify-self-start">ТОВАР </div>

            <div className="justify-self-center">ЦІНА </div>

            <div className="justify-self-center">КІЛЬКІСТЬ</div>

            <div className="justify-self-end">ДО СПЛАТИ </div>

        </div>

        {order.products && order.products.map(item => {

            return <OrderItem key={item.id} item={item}></OrderItem>

        })}
    
```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						65

```

        </div>

    </div> );

};

export default OrderDetails;

```

Додаток В

Реалізація функції, що відповідає за перегляд замовлень клієнта

Лістинг OrderClient.tsx

```

"use client";

import { Order, Product, User } from "@prisma/client";
import React, { useCallback } from "react";
import { DataGrid, GridColDef } from '@mui/x-data-grid';
import { formatPrice } from "@utils/formatPrice";
import Heading from "@app/components/Heading";
import Status from "@app/components/Status";
import { MdAccessTimeFilled, MdDeliveryDining, MdDone, MdRemoveRedEye } from "react-icons/md";
import ActionBtn from "@app/components/ActionBtn";
import axios from "axios";
import toast from "react-hot-toast";
import { useRouter } from "next/navigation";
import moment from "moment";

interface OrdersClientProps {
    orders: ExtendedOrder[]
}

type ExtendedOrder = Order & {
    user: User
}

const OrdersClient: React.FC<OrdersClientProps> = ({ orders }) => {

```

```

const router = useRouter();

let rows: any = []

if (orders) {
  rows = orders.map((order) => {
    return {
      id: order.id,
      customer: order.user.name,
      amount: formatPrice(order.amount / 100),
      paymentStatus: order.status,
      date: moment(order.createdAt).fromNow(),
      deliveryStatus: order.deliveryStatus,
    }
  })
}

const columns: GridColDef[] = [
  {field: "id", headerName: "ID", width: 220},
  {field: "customer", headerName: "Твій нікнейм", width: 130},
  {field: "amount", headerName: "Ціна", width: 100, renderCell: (params) => {
    return(<div className="font-bold text-slate-800">{params.row.amount}</div>);
  },},
  {field: "paymentStatus", headerName: "Статус оплати", width: 130, renderCell: (params) => {
    return(<div>{params.row.paymentStatus === 'в очікуванні' ? ( <Status text="в очікуванні" icon={MdAccessTimeFilled} bg="bg-slate-200" color="text-slate-700"/> ) :
      params.row.paymentStatus === "complete" ? ( <Status text="сплачено" icon={MdDone} bg="bg-green-200" color="text-green-700"/> ) : <></>
    }</div>);
  },},
  {field: "deliveryStatus", headerName: "Статус доставки", width: 130, renderCell: (params)
=> {
    return(<div>{params.row.deliveryStatus === 'в очікуванні' ? ( <Status text="в очікуванні" icon={MdAccessTimeFilled} bg="bg-slate-200" color="text-slate-700"/> ) :

```

```

        params.row.deliveryStatus === "відправлено" ? ( <Status text="відправлено"
icon={MdDeliveryDining} bg="bg-purple-200" color="text-purple-700"/>) : params.row.deliveryStatus
=== "доставлено" ?

        ( <Status text="доставлено" icon={MdDone} bg="bg-green-200" color="text-green-700"/>)
: <></>

    </div>);

  },},

  {

    field: "date", headerName: "Час замовлення", width: 130,

  },

  {field: "action", headerName: "Інструменти", width: 200, renderCell: (params) => {

    return(<div className="flex justify-between gap-4 w-full">

      <ActionBtn icon={MdRemoveRedEye} onClick={() => {

        router.push(`/order/${params.row.id}`);

      }}/>

    </div>);

  },},

];

return (

<div className="max-w-[1150px] m-auto text-xl">

  <div className="mb-4 mt-8">

    <Heading title=" Твої замовлення" center/>

  </div>

  <div style={{height: 600, width: "100%"}}>

    <DataGrid

      rows={rows}

      columns={columns}

      initialState={{

        pagination: {

          paginationModel: { page: 0, pageSize: 7 },

        },

      }}

    >

```

```

        pageSizeOptions={[7, 20]}

        checkboxSelection

        disableRowSelectionOnClick
    />
</div>
</div>);
}

```

```
export default OrdersClient
```

Додаток Г

Реалізація функції, що відповідає за перегляд статистики інтернет магазину

Лістинг Summary.tsx

```

"use client";

import { Order, Product, User } from "@prisma/client";
import React, { useEffect, useState } from "react";
import Heading from "../components/Heading";
import { formatPrice } from "@utils/formatPrice";
import { formatNumber } from "@utils/formatNumber";

interface SummaryProps{
    orders: Order[];
    products: Product[];
    users: User[];
}

type SummaryDataType = {
    [key: string]:{
        label: string;
        digit: number;
    }
}

```

```

    }
  }

const Summary:React.FC<SummaryProps> = ({orders, products, users}) => {
  const [summaryData, setSummaryData] = useState<SummaryDataType>({
    sale:{
      label: "Товару продана на суму",
      digit: 0
    },
    products:{
      label: "Всього унікальних товарів",
      digit: 0
    },
    orders:{
      label: "Всього замовлень",
      digit: 0
    },
    paidOrders:{
      label: "Сплачених замовлень",
      digit: 0
    },
    unpaidOrders:{
      label: "Не сплачених замовлень",
      digit: 0
    },
    users:{
      label: "Всього користувачів",
      digit: 0
    },
  })

  useEffect(()=>{
    setSummaryData ((prev) => {

```

```

let tempData = {...prev}

const totalSale = orders.reduce((acc,item) => {
    if(item.status === 'complete'){
        return acc + item.amount
    } else return acc
}, 0)

const paidOrders = orders.filter((order => {
    return order.status === 'complete'
}))

const unpaidOrders = orders.filter((order => {
    return order.status === 'в очікуванні'
}))

tempData.sale.digit = totalSale;
tempData.orders.digit = orders.length;
tempData.paidOrders.digit = paidOrders.length;
tempData.unpaidOrders.digit = unpaidOrders.length;
tempData.products.digit = products.length;
tempData.users.digit = users.length;

return tempData
})
},[orders, products, users])

const summaryKeys = Object.keys(summaryData)

return ( <div className="max-w-[1150px m-auto]">
    <div className="mb-4 mt-8">
        <Heading title="Статистика" center/>

```

```

</div>

<div className="grid grid-cols-2 gap-3 max-h-50vh overflow-y-auto">
  {
    summaryKeys && summaryKeys.map((key) => {
      return <div key={key} className="rounded-xl border-2 p-4 flex flex-col
items-center gap-2 transition">
        <div className="text-xl md:text-4xl font-bold">
          {
            summaryData[key].label === 'Товару продана на суму' ?
<>{formatPrice(summaryData[key].digit / 100)}</> : <>{formatNumber(summaryData[key].digit)}</>
          }
        </div>
        <div>{summaryData[key].label}</div>
      </div>
    })
  }
</div>
</div> );
}

export default Summary;

```