

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 лижного спорядження»

Виконав студент групи ВТ-22-1
КОРОЦІНСЬКИЙ Віктор Вікторович

Керівник роботи:
КОНЦИДАЙЛО Андрій Михайлович

Рецензент:
ВЛАСЕНКО Олег Васильович

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

«16» липня 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КОРОЦІНСЬКИЙ Віктор Вікторович**

Керівник роботи: **КОНЦИДАЙЛО Андрій Михайлович**

Тема роботи: «Інтернет-магазин лижного спорядження»,

затверджена Наказом закладу вищої освіти від «16» липня 2026 р., №396/с

Вихідні дані для роботи: **використання веборієнтованих технологій для розробки інформаційно-керуючої системи інтернет-магазину гірськолижного спорядження «SkiZone» з метою автоматизації процесів ведення каталогу товарів, обліку замовлень, інтеграції персоналізованих сервісів клієнта та оптимізації взаємодії між покупцем і адміністратором системи.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Концидайло А.М.	16.07.2026р.	16.07.2026р.
2	Концидайло А.М.	16.07.2026р.	16.07.2026р.
3	Концидайло А.М.	16.07.2026р.	16.07.2026р.

Реферат

Пояснювальна записка до кваліфікаційної роботи на тему «Інтернет-магазин лижного спорядження» складається з реферату, вступу, трьох розділів, висновків, списку використаних джерел та додатків. Пояснювальна записка містить 3 розділи, 73 сторінок, 38 рисунків, 8 таблиць, список із 30 використаних джерел та додатки.

Метою роботи є проектування та розробка повнофункціонального інтернет-магазину лижного спорядження «SkiZone», що забезпечує автоматизацію процесу вибору та купівлі товарів для зимових видів спорту, управління замовленнями, фінансовий контроль, а також підтримує безпечну автентифікацію користувачів та адміністрування контенту.

Об'єктом дослідження є процес електронної комерції та онлайн-продажів у сфері спеціалізованого спортивного обладнання, зокрема процеси каталогізації товарів, взаємодії з клієнтами та обробки замовлень.

Предметом дослідження є архітектурні методи, інформаційні моделі, сучасні веборієнтовані технології, алгоритми та програмні засоби автоматизації роботи інтернет-магазину (каталог, кошик, список бажань, авторизація й адміністрування).

У роботі застосовано методи об'єктно-орієнтованого проектування та програмування, клієнт-серверної архітектури вебдодатків, компонентного підходу на базі сучасних фреймворків, технології нереляційного (документо-орієнтованого) моделювання даних, механізми безпечної токенної автентифікації (JWT) та криптографічного хешування паролів.

КЛЮЧОВІ СЛОВА: ВЕБСИСТЕМА, ЕЛЕКТРОННА КОМЕРЦІЯ, ІНТЕРНЕТ-МАГАЗИН, NEXT.JS, NODE.JS.

					ІПЗ.КР.Б – 121 – 26 – ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата	Інтернет-магазин лижного спорядження	Літ.		Арк.	Аркушів		
Розроб.		Короцінський В.В.									
Керівник		Корнцидайло А.М.						3		73	
						Житомирська політехніка, група ВТ-22-І					
Рецензент		Власенко О.В.									
Зав. Каф.		Вакалюк Т.А.									

ABSTRACT

The explanatory note to the qualification work on the topic "Online Ski Equipment Store" consists of an abstract, an introduction, three sections, conclusions, a list of references and appendices. The explanatory note contains 3 sections, 73 pages, 38 figures, 8 tables, a list of 30 references and appendices.

The purpose of the work is to design and develop a fully functional online ski equipment store "SkiZone" that automates the process of selecting and purchasing winter sports goods, order management, financial control, and supports secure user authentication and content administration.

The object of research is the process of e-commerce and online sales in the field of specialized sports equipment, in particular the processes of product cataloguing, customer interaction and order processing.

The subject of research is architectural methods, information models, modern web-oriented technologies, algorithms and software tools for automating the online store operation (catalog, shopping cart, wishlist, authentication and administration).

The work applies methods of object-oriented design and programming, client-server architecture of web applications, component-based approach using modern frameworks, non-relational (document-oriented) data modeling technology, secure token-based authentication mechanisms (JWT) and cryptographic password hashing.

KEYWORDS: E-COMMERCE, NEXT.JS, NODE.JS, ONLINE STORE, WEB SYSTEM.

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ВИЗНАЧЕННЯ ФУНКЦІОНАЛУ.....	9
1.1. Опис предметної галузі та постановка задачі.....	9
1.2. Аналіз аналогічних розробок.....	12
1.3. Вибір технологічного стеку та інструментальних засобів.....	17
1.4. Визначення функціональних вимог до системи.....	21
Висновки до першого розділу.....	22
РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ЛИЖНОГО ОБЛАДНАННЯ.....	25
2.1. Визначення варіантів використання системи.....	25
2.2. Розробка бази даних системи.....	30
2.3. Проектування алгоритмів роботи системи.....	32
2.4. Реалізація серверної частини системи.....	39
Висновки до другого розділу.....	48
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ІНТЕРНЕТ- МАГАЗИНУ ЛИЖНОГО ОБЛАДНАННЯ.....	49
3.1. Порядок встановлення та налаштування системи.....	49
3.2. Опис інтерфейсу користувача.....	51
3.3. Тестування програмного комплексу.....	58
Висновки до третього розділу.....	61
ВИСНОВКИ.....	63
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	66
ДОДАТКИ.....	69

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

ЗВО – заклад вищої освіти

ІТ – інформаційні технології

ПЗ – програмне забезпечення

СУБД – система управління базами даних

API – інтерфейс прикладного програмування (Application Programming Interface)

BSON – бінарний формат збереження документів у NoSQL БД (Binary JSON)

CRUD – базові операції роботи з даними: створення, читання, оновлення, видалення (Create, Read, Update, Delete)

HTTP/HTTPS – протокол передачі гіпертексту (захищений)

JSON – текстовий формат обміну даними

JWT – компактний стандарт для безпечної передачі маркерів доступу (JSON Web Token)

MVC – архітектурний шаблон побудови програмного забезпечення, що розділяє застосунок на модель (Model), представлення (View) та контролер (Controller)

NoSQL – клас СУБД, що реалізують нереляційні моделі даних

ODM – бібліотека для об'єктно-документного відображення (Object Document Mapper)

REST/RESTful API – архітектурний стиль взаємодії компонентів через протокол HTTP

RFC – офіційний технічний стандарт Інтернету

SEO – пошукова оптимізація вебсторінок (Search Engine Optimization)

SSR – серверний рендеринг сторінок (Server-Side Rendering)

UI/UX – користувацький інтерфейс та досвід взаємодії (User Interface / User Experience)

URL – уніфікований покажчик ресурсу в мережі

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Стрімкий розвиток електронної комерції в Україні та зростання популярності зимових видів спорту формують стабільний попит на якісне лижне спорядження. Проте більшість існуючих спеціалізованих магазинів досі орієнтовані на офлайн-продажі або використовують застарілі технічні рішення, що обмежують швидкість обробки замовлень, синхронізацію складських залишків та якість обслуговування клієнтів.

Сучасна випускна кваліфікаційна робота бакалавра з інженерії програмного забезпечення має демонструвати здатність розробника створювати високоефективні, безпечні та адаптивні вебплатформи. Тому проектування інтернет-магазину лижного спорядження «SkiZone» є надзвичайно актуальним, адже воно поєднує оптимізацію інтерфейсу користувача (UI/UX), надійність збереження даних, високу швидкість обробки сезонних пікових навантажень та використання сучасних стандартів захисту персональних даних.

Метою роботи є проектування та розробка масштабованого інтернет-магазину лижного спорядження для автоматизації процесів онлайн-торгівлі, ведення інтерактивного каталогу, управління кошиком і списком бажань, а також адміністрування товарних позицій за умов забезпечення надійної JWT-автентифікації.

Для досягнення поставленої мети необхідно виконати такі **завдання**: Проаналізувати предметну галузь електронної комерції та особливості ринку лижного спорядження, дослідити наявні аналоги та сформулювати конкурентні переваги системи. Сформулювати функціональні, нефункціональні та системні вимоги до програмного продукту. Обґрунтувати вибір сучасного технологічного стеку (Next.js, Node.js, Express.js, MongoDB) та спроектувати архітектуру взаємодії клієнтської і серверної частин додатка. Спроектувати схему бази даних (NoSQL BSON-документи) та розробити UML-діаграми варіантів використання, класів і послідовності. Реалізувати серверне REST API з інтеграцією механізмів автентифікації та захисту даних. Розробити

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

клієнтську частину вебдодатка з адаптивним інтерфейсом, оптимізацією зображень та інтеграцією з API.

Провести тестування розробленої системи (створити протокол тестування) та проаналізувати отримані результати.

Об'єктом дослідження є процеси електронної комерції, онлайн-продажів та адміністрування спеціалізованих вебплатформ у сфері роздрібної торгівлі спортивними товарами.

Предметом дослідження є архітектурні методи, інформаційні моделі, сучасні веборієнтовані технології (Next.js, Node.js), алгоритми захисту даних та програмні засоби автоматизації каталогізації, онлайн-замовлень і комунікації з клієнтами інтернет-магазину.

Практичне значення результатів полягає у створенні готового до розгортання програмного комплексу (інтернет-магазину SkiZone), архітектурні рішення та модульна структура якого можуть бути використані як інженерна основа для реальних e-commerce платформ. Гнучкість NoSQL бази даних та використання RESTful API дозволяють легко масштабувати систему та інтегрувати її в майбутньому із зовнішніми платіжними шлюзами (еквайрингом), сервісами доставки та CRM-системами обліку.

Структура та обсяг роботи. Пояснювальна записка складається з реферату, вступу, 3-х розділів, висновків, списку використаних джерел із 30 найменувань та додатків. Робота містить 8 таблиць та 33 ілюстрацій. Повний обсяг записки становить 68 сторінок.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ВИЗНАЧЕННЯ ФУНКЦІОНАЛУ

1.1 Опис предметної галузі та постановка задачі

Лижне обладнання являє собою комплекс спеціалізованих елементів, розроблених із застосуванням новітніх матеріалів і технологій для забезпечення оптимальних умов занять лижним спортом. Основним компонентом цієї системи є лижі, виконані з композитних матеріалів, що характеризуються високою жорсткістю та низькою вагою. Конструкція лиж включає спеціальне покриття, яке покращує ковзання та зменшує тертя з поверхнею, а також оптимізує розподіл ваги для забезпечення стабільності під час спуску [1].

Система кріплень є невід'ємною частиною лижного спорядження. Вона забезпечує надійне з'єднання лиж із черевиками, ефективно передає сили, що виникають під час руху, та має механізми безпеки, які запобігають травмам за рахунок своєчасного роз'єднання у разі надмірного навантаження. Лижні черевики повинні забезпечувати максимальну підтримку стопи і щиколотки, сприяючи коректній передачі енергії від спортсмена до лиж. Їх конструкція базується на використанні амортизуючих вставок і систем регулювання, що дозволяють адаптувати спорядження до індивідуальних анатомічних особливостей.

До складу лижного спорядження також входять: лижні палиці з легких та міцних матеріалів, що сприяють стабілізації руху; лижні окуляри, які забезпечують захист очей від ультрафіолетових променів та снігових відблисків; спеціалізований одяг для захисту від морозу та вітру. Завдяки постійному вдосконаленню технологій виробництва, сучасне лижне обладнання демонструє високий рівень ефективності та безпеки [2].

Зростання популярності зимових видів спорту в Україні та Європі формує стійкий попит на якісне спорядження. Водночас більшість спеціалізованих магазинів представлені лише офлайн або мають технічно застарілі вебсайти.

Тому створення профільної вебплатформи для автоматизації продажів та

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

централізованого управління товарними позиціями є необхідним кроком для вирішення цих технічних обмежень

Це зумовлює потребу у впровадженні сучасних рішень для електронної комерції у сфері спортивного ритейлу. Для автоматизації процесів продажу та покращення взаємодії з клієнтами проєктована система «SkiZone» має реалізовувати комплексний функціонал, розподілений за відповідними архітектурними компонентами.

Взаємодія користувача з платформою починається з модуля каталогу товарів, який забезпечує повноцінне відображення наявного асортименту з можливістю фільтрації за категоріями, повнотекстовим пошуком за назвою та детальним описом, а також динамічним сортуванням за різними критеріями. Безпеку персональних даних та розмежування прав доступу покладено на модуль авторизації та реєстрації, що підтримує рольову модель із розподілом на ролі «користувач» та «адміністратор» та здійснює автентифікацію через захищені JWT-токени терміном дії в одну добу.

Безпосередній процес здійснення покупок автоматизовано завдяки модулю кошика, який підтримує логіку додавання товарів з автоматичним збільшенням кількості при повторному обранні однієї й тієї ж позиції, а також забезпечує механізм фінального оформлення замовлення. Додатковим інструментом для підвищення лояльності покупців є модуль списку бажань, призначений для збереження товарів для майбутньої покупки з можливістю повного чи часткового очищення переліку. Індивідуалізація взаємодії з клієнтом реалізована через модуль особистого кабінету, де користувач може редагувати власні облікові дані та переглядати історію замовлень. Для централізованого управління контентом та операційною діяльністю передбачено адміністративну панель, що надає інструментарій для виконання повного циклу CRUD-операцій над товарами із завантаженням фотографій, а також інтерфейс для управління поточними замовленнями.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

Етапи розробки інтернет-магазину SkiZone

№	Назва етапу	Результат
1	Аналіз предметної галузі та вимог	Перелік вимог, порівняльна таблиця аналогів
2	Проектування архітектури та БД	UML-діаграми, схема MongoDB
3	Розробка серверної частини (REST API)	Бекенд на Node.js + Express
4	Розробка клієнтської частини	Вебдодаток на Next.js
5	Тестування та налагодження	Протокол тестування

Ринок електронної комерції в Україні демонструє стрімке зростання попри складні економічні умови. За даними Української асоціації директ маркетингу, обсяг онлайн-торгівлі щороку збільшується на 25–30%, а кількість активних онлайн-покупців перевищила 13 мільйонів осіб. Частка мобільної комерції складає понад 60% від загального обсягу онлайн-продажів, що свідчить про необхідність адаптивного дизайну для будь-якого сучасного інтернет-магазину.

Ринок лижного спорядження має виражену сезонність — попит концентрується у період з жовтня по березень і в цей час зростає у 3–4 рази порівняно з іншими сезонами. Дослідження поведінки споживачів показують, що понад 70% покупців попередньо вивчають товар онлайн перед придбанням, порівнюючи характеристики та ціни у різних магазинах. Це формує чіткий запит на зручний спеціалізований інтернет-магазин із детальними описами товарів, якісними фотографіями та простим процесом оформлення замовлення.

Аналіз актуальних тенденцій ринку дозволяє виділити ключові вимоги до сучасного інтернет-магазину спортивного обладнання. По-перше, зручний пошук та фільтрація товарів є критично важливими — користувач має знайти потрібне спорядження за лічені секунди. По-друге, збереження списку бажань між сесіями дозволяє покупцю повертатися до обраних товарів

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

пізніше, що суттєво підвищує конверсію. По-третє, простий процес оформлення замовлення з мінімальною кількістю кроків зменшує відсоток покинутих кошиків. По-четверте, підтримка кількох способів оплати — готівкою та карткою — задовольняє потреби різних категорій покупців. Саме ці вимоги і стали основою при проектуванні системи SkiZone.

1.2 Аналіз аналогічних розробок

Перед розробкою власного рішення проведено аналіз існуючих інтернет-магазинів спортивного обладнання з метою визначення переваг та недоліків конкурентів і формування оптимального набору функцій власної системи.

Extremstyle (extremstyle.ua) — сучасна платформа електронної комерції, орієнтована на продаж спортивного одягу та спорядження [3]. Система використовує багаторівневу архітектуру з чітким розділенням фронтенду та бекенду. Відзначається високою модульністю, що дозволяє окремо розробляти та тестувати різні функціональні блоки. Однією з переваг є гнучкість у налаштуванні бази даних та використання RESTful API для забезпечення безперебійної взаємодії між клієнтом і сервером.

Extremstyle відзначається модульною архітектурою, що суттєвоспрощує процес тестування окремих компонентів та їх незалежного розгортання. Використання RESTful API забезпечує ефективну взаємодію між клієнтською та серверною частинами, а подієво-орієнтована модель обробки запитів забезпечує стабільну роботу платформи навіть при значних навантаженнях. Разом із тим, платформа має і певні недоліки: інтеграція сторонніх сервісів є технічно складною та потребує значних зусиль, а при суттєвому зростанні навантаження можуть виникати труднощі з горизонтальним масштабуванням системи. Отже, архітектурна модель даного ресурсу демонструє ефективну роботу в межах стандартних REST-запитів, проте має обмеження при інтенсивному масштабуванні серверних потужностей.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

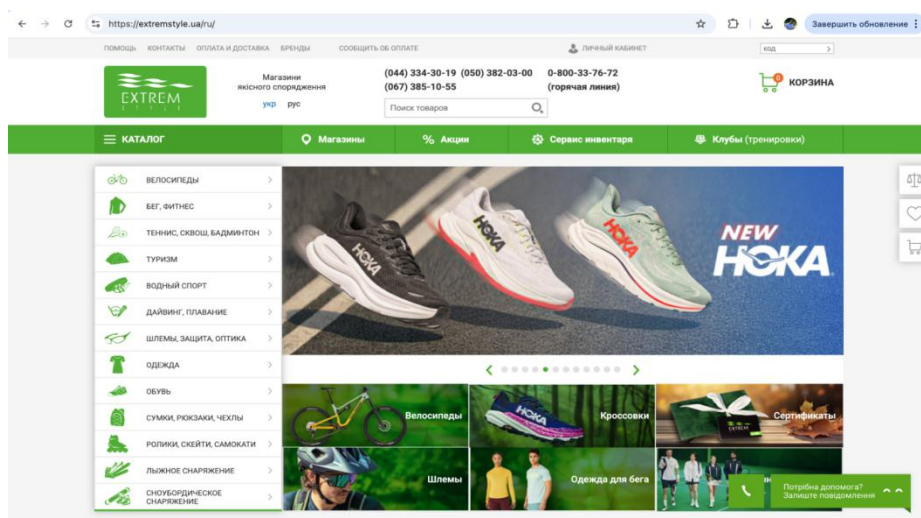


Рис. 1.1 – Головна сторінка Extremstyle

Unisport (unisport.ua) — комплексна платформа для продажу спортивних товарів [4]. Відзначається інноваційним підходом до розробки інтерфейсу та використанням сучасних технологій: React для клієнтської частини і Node.js із Express для серверної логіки. Основними перевагами є зрозуміла та добре документована архітектура, а також використання сучасних засобів безпеки

Серед недоліків — проблеми з продуктивністю через залежність від зовнішніх API та ресурсомісткі операції.

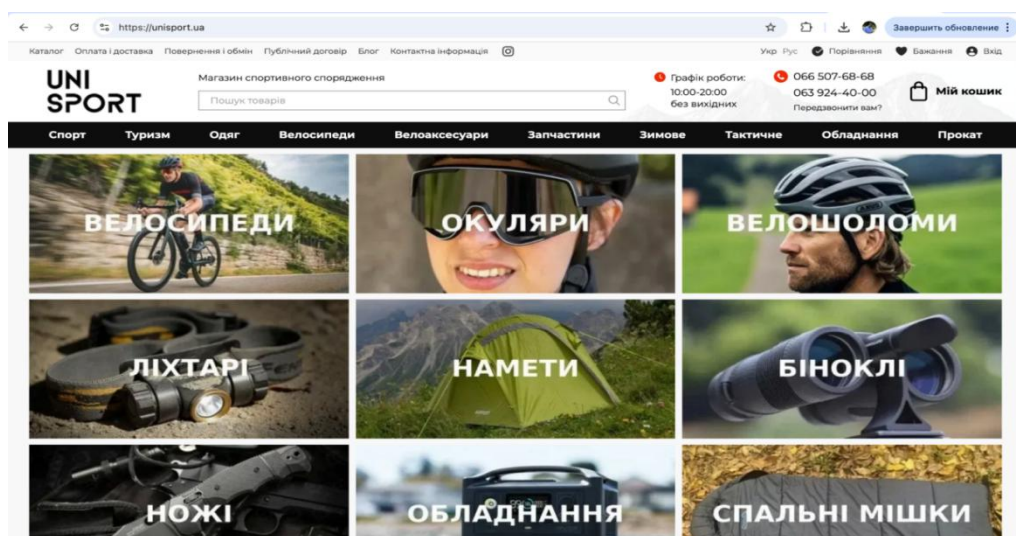


Рис. 1.2 – Головна сторінка Unisport

Keeperstore (keeperstore.ua) — інтернет-магазин спортивного спорядження [5]. Система базується на інтеграції інтерфейсних технологій та серверної логіки для забезпечення інтуїтивного UX/UI. Проте суттєвим недоліком платформи є жорстка структура, яка обмежує гнучке налаштування модулів при впровадженні нових функцій. З інженерної точки зору, система демонструє зниження швидкодії на мобільних пристроях через перевантаженість фронтенду важкими скриптами. Крім того, в періоди пікових сезонних навантажень виникають затримки синхронізації складських залишків у базі даних із каталогом сайту. Також обмежений функціонал особистого кабінету й відсутність автономного списку бажань (Wishlist) змушують користувачів координувати замовлення з менеджерами в ручному режимі.

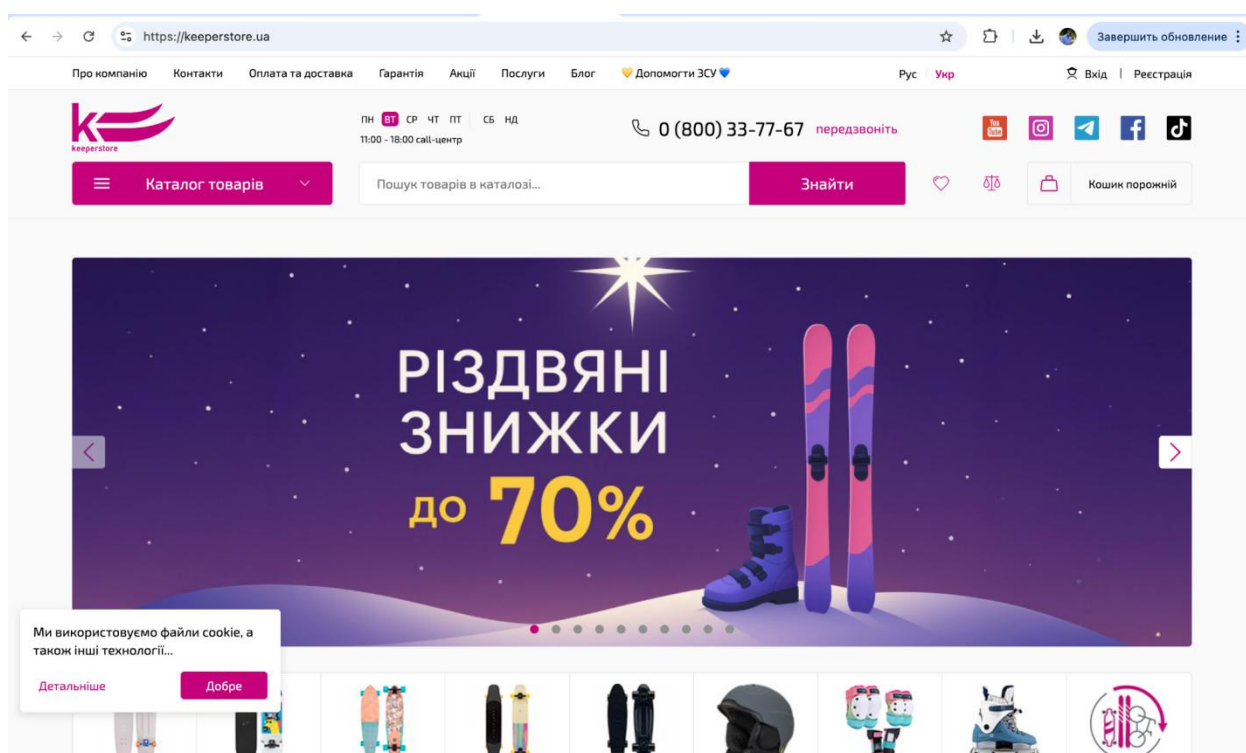


Рис. 1.3 – Головна сторінка Keeperstore

Decathlon.ua — офіційний сайт відомого французького спортивного ритейлера в Україні, що пропонує широкий асортимент товарів для зимових видів спорту власних торгових марок [7]. Платформа відзначається зручною навігацією та детальними описами товарів з технічними характеристиками. Особливістю є система фільтрації за рівнем підготовки користувача —

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

початківець, досвідчений, експерт — що суттєво спрощує вибір підходящого спорядження. Сайт також містить відео-огляди товарів та рекомендації щодо вибору спорядження залежно від стилю катання та фізичних параметрів спортсмена. Серед переваг платформи слід виділити широкий асортимент власних торгових марок за доступними цінами, розвинену систему відгуків та оцінок покупців і наявність мобільного додатку. До недоліків належать відсутність спеціалізації виключно на лижному обладнанні, обмежений вибір преміальних брендів та відсутність функції списку бажань з можливістю очищення.

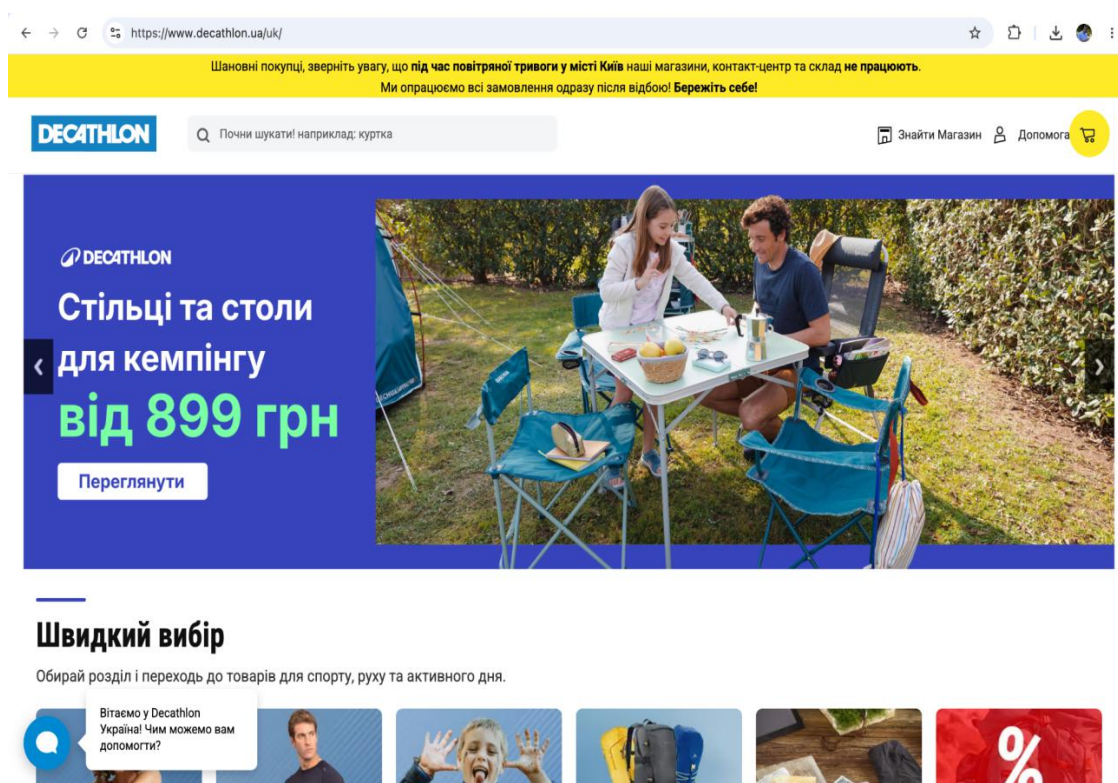


Рис. 1.4 – Головна сторінка Decathlon

Rozetka.ua є найбільшим українським маркетплейсом та лідером у сфері вітчизняної електронної комерції [6]. До ключових переваг платформи належать масштабний асортимент товарів, інформативні картки продукції з відгуками користувачів, розвинена система багатокритеріальної фільтрації, а також підтримка кількох зручних способів оплати.

Водночас прагнення охопити всі категорії товарів зумовлює певні недоліки. Головним мінусом є повна відсутність вузької спеціалізації на професійному лижному та сноубордичному обладнанні, що обмежує

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

можливості точного підбору спорядження за специфічними технічними характеристиками. Крім того, через велику кількість різноманітного контенту інтерфейс платформи є суттєво перевантаженим, що ускладнює швидку навігацію та пошук спеціалізованих спортивних товарів.

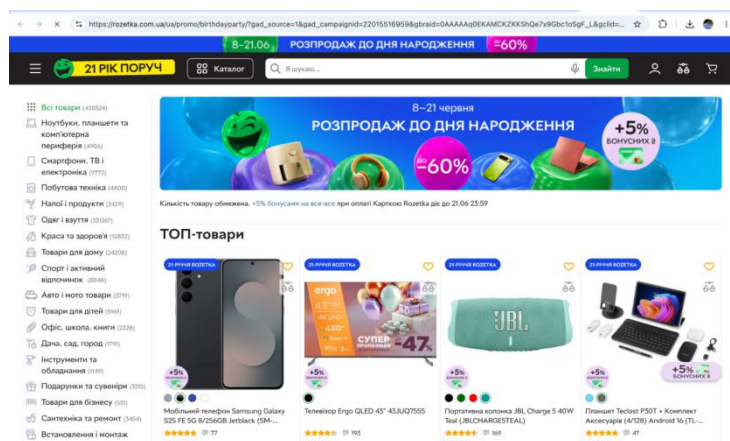


Рис. 1.5 – Головна сторінка Rozetka

Табл. 1.2

Порівняльна характеристика аналогів програмного продукту

Характеристика	Extremstyle	Unisport	Keeperstore	Decathlon	SkiZone
Спеціалізація на лижах	±	±	±	—	+
Кошик та замовлення	+	+	+	+	+
Список бажань	—	—	—	+	+
Особистий кабінет	+	+	±	+	+
JWT-аутентифікація	±	+	±	—	+
Адміністративна панель	+	+	+	+	+
Оплата карткою	+	+	+	+	+
Сучасний дизайн	+	+	±	+	+
Очищення вішліста	-	-	-	-	+
Фільтр за рівнем підготовки	-	-	-	+	-

Проведений аналіз показав, що жоден з розглянутих аналогів не поєднує одночасно вузьку спеціалізацію на лижному обладнанні, сучасну JWT-аутентифікацію, список бажань з функцією повного очищення та сучасний адаптивний дизайн. Великі маркетплейси на зразок Rozetka.ua

мають широкий функціонал, проте не спеціалізуються на лижному спорядженні. Спеціалізовані магазини такі як Girski.ua орієнтовані на правильну товарну нішу, проте технічно застарілі. Розроблювана система SkiZone усуває ці недоліки, поєднуючи спеціалізацію на лижному обладнанні з повним набором сучасних функцій e-commerce.

1.3 Вибір технологічного стеку та інструментальних засобів

Вибір технологічного стеку є одним з ключових рішень при розробці вебдодатку, оскільки безпосередньо впливає на продуктивність, масштабованість та підтримуваність системи. При виборі технологій для реалізації інтернет-магазину SkiZone враховувалися такі критерії: продуктивність при обробці великої кількості одночасних запитів, активна підтримка спільноти розробників, наявність необхідних бібліотек та інструментів, а також простота розгортання та налаштування [5].

Вебдодаток побудований на клієнт-серверній архітектурі, де фронтенд та бекенд є незалежними компонентами, що взаємодіють через RESTful API[29]. Такий підхід забезпечує чітке розмежування відповідальності між частинами системи, спрощує масштабування та дозволяє незалежно розробляти і тестувати кожен компонент.

Next.js версії 15 є React-фреймворком від компанії Vercel, що надає додаткові можливості поверх бібліотеки React [8]. Ключовою особливістю є підтримка серверного рендерингу (SSR), що суттєво покращує SEO-оптимізацію та швидкість першого завантаження сторінки. Файлова маршрутизація через директорію src/pages/ спрощує організацію сторінок — кожен файл автоматично стає маршрутом без додаткового налаштування. Вбудована оптимізація зображень через компонент Image автоматично конвертує фотографії товарів у формат WebP та адаптує їх розмір під пристрій користувача, що критично важливо для сторінок каталогу з великою кількістю зображень.

Node.js версії 18 є платформою для виконання JavaScript на стороні сервера, що базується на рушії V8 браузера Chrome [9]. Платформа використовує подієво-орієнтовану неблокуючу модель вводу-виводу з однопотоковим циклом подій (Event Loop), що дозволяє ефективно обробляти тисячі одночасних з'єднань без блокування потоку виконання. Це особливо важливо для інтернет-магазину, де в пікові сезонні моменти можуть одночасно працювати сотні користувачів. Додатковою перевагою є використання JavaScript як єдиної мови програмування для клієнтської та серверної частин, що спрощує розробку та обмін кодом між частинами системи.

Express.js версії 4 є мінімалістичним та гнучким фреймворком для розробки вебдодатків на Node.js [10]. Він реалізує концепцію middleware-ланцюжка, де кожен HTTP-запит послідовно проходить через функції обробки. У проекті SkiZone використовуються такі middleware: cors() для дозволу крос-доменних запитів між фронтендом та бекендом, express.json() для парсингу тіла запитів у форматі JSON, authMiddleware для перевірки JWT-токена на захищених маршрутах, adminMiddleware для перевірки прав адміністратора та upload.single("image") для обробки завантаження фотографій товарів.

MongoDB версії 6 є документо-орієнтованою СУБД класу NoSQL, де дані зберігаються у форматі BSON — бінарному розширенні JSON [11]. На відміну від реляційних СУБД, MongoDB дозволяє зберігати складні вкладені структури без необхідності нормалізації, що особливо зручно для зберігання кошика з масивом товарів або замовлення з деталями доставки. Mongoose версії 7 є ODM-бібліотекою для MongoDB, що надає зручний інтерфейс для визначення схем даних, валідації полів та побудови запитів [12]. Метод populate() автоматично замінює ObjectId-посилання на повні документи з пов'язаних колекцій, що спрощує отримання даних кошика разом з інформацією про товари.

Бібліотека `bcryptjs` використовується для безпечного зберігання паролів користувачів [13]. При реєстрації пароль хешується функцією `bcrypt.hash(password, 10)`, де 10 — кількість раундів соління (salt rounds). Отриманий хеш зберігається у базі даних замість відкритого паролю, що унеможлиблює його відновлення навіть у разі витоку бази даних. При авторизації виконується `bcrypt.compare(password, hash)` для порівняння введеного паролю з хешем.

JSON Web Token (JWT) — відкритий стандарт RFC 7519 для передачі інформації між сторонами у вигляді JSON-об'єкта [14]. У системі SkiZone JWT використовується для аутентифікації користувачів. Після успішного входу сервер генерує токен з `payload { userId: user._id, role: user.role }`, підписує його секретним ключем `JWT_SECRET` та встановлює термін дії 1 доба (`expiresIn: "1d"`). Клієнт зберігає токен у `localStorage` та передає його у заголовок кожного запиту до захищених маршрутів у форматі `Authorization: Bearer <token>`. Middleware `authMiddleware.js` верифікує токен через `jwt.verify()` та при успіху зберігає `payload` у `req.user`.

Multer є middleware для Node.js, що забезпечує обробку `multipart/form-data` запитів, які використовуються для завантаження файлів. У проекті Multer налаштований на збереження фотографій товарів у директорії `uploads/` з іменем файлу у форматі `Date.now() + "-" + random + extension` [15]. Filefilter обмежує дозволені типи файлів лише зображеннями `jpg`, `jpeg`, `png` та `gif`, що запобігає завантаженню шкідливих файлів. Завантажені зображення роздаються як статичні файли через `app.use("/uploads", express.static("uploads"))` у `server.js`.

Axios є HTTP-клієнтом на основі Promise для браузера та Node.js [16]. У клієнтській частині SkiZone Axios використовується для надсилання всіх HTTP-запитів до серверного API. Перевагами є автоматичне перетворення JSON-даних, зручний інтерфейс для встановлення заголовків (зокрема `Authorization` з JWT-токеном) та обробки помилок через блоки `try/catch`.

Git та GitHub використовуються для контролю версій коду проекту [17]. Репозиторій містить обидві частини проекту у директоріях skizone-backend та skizone-frontend. Використання системи контролю версій дозволяє відслідковувати історію змін, повертатися до попередніх версій та зберігати код у хмарному репозиторії.

Visual Studio Code використовується як основне середовище розробки завдяки підтримці JavaScript та TypeScript, вбудованому терміналу, розширенням для роботи з MongoDB Compass та зручному дебаггеру Node.js [18].

Табл. 1.3

Порівняльна характеристика засобів розробки системи SkiZone

Технологія	Версія	Призначення	Ключова перевага
Next.js	15.x	Фронтенд фреймворк	SSR, файлова маршрутизація, оптимізація зображень
Node.js	18+	Серверна платформа	Асинхронна обробка, єдина мова з фронтендом
Express.js	4.x	Бекенд фреймворк	Гнучкість, middleware, REST API
MongoDB	6+	СУБД (NoSQL)	Гнучка схема, швидкість, BSON-документи
Mongoose	7.x	ODM-бібліотека	Схеми, валідація, populate()
JWT	–	Автентифікація	Без сесій, payload: {userId, role}, термін 1d
bcryptjs	–	Хешування паролів	Алгоритм bcrypt, salt rounds: 10
Multer	–	Завантаження фото	diskStorage, фільтр jpg/png/gif
Axios	–	HTTP-запити на клієнті	Promise-based, зручний інтерфейс

1.4 Визначення функціональних вимог до системи

Функціональні та нефункціональні вимоги до системи сформовано на основі аналізу предметної галузі, вивчення аналогів та потреб цільової аудиторії інтернет-магазину лижного обладнання. Вимоги визначають що саме має робити система та які обмеження на неї накладаються [19].

Щодо функціональних вимог система повинна забезпечувати реєстрацію нових користувачів з хешуванням пароллю через алгоритм bcrypt з кількістю раундів соління рівним 10, а також авторизацію через форму введення логіну та пароллю з генерацією JWT-токена терміном дії 1 доба. Система має надавати можливість перегляду каталогу товарів з пошуком одночасно за полями назви та опису товару, фільтрацією за категорією та сортуванням за різними критеріями. Авторизовані користувачі повинні мати можливість додавати товари до кошика, причому при повторному додаванні того самого товару його кількість має автоматично збільшуватися на одиницю без створення дублікату. Система підтримує два способи оплати замовлення — готівкою при доставці та банківською картою з валідацією номера, терміну дії та CVV-коду. Авторизовані користувачі повинні мати доступ до списку бажань з можливістю додавання товарів, їх перегляду та повного очищення списку одним натисканням. В особистому кабінеті користувач має можливість редагувати логін та пароль, а також переглядати історію своїх замовлень з поточними статусами. Адміністратор системи повинен мати доступ до повного набору CRUD-операцій над товарами із завантаженням фотографій, а також до перегляду всіх замовлень від усіх користувачів з можливістю зміни їх статусу.

Нефункціональні вимоги визначають якісні характеристики системи. Час відповіді API на запити користувача не повинен перевищувати 3 секунди за нормальних умов навантаження. Всі захищені маршрути API перевіряють наявність та валідність JWT-токена через заголовок Authorization у форматі Bearer token, а при невалідному токені повертають код відповіді 401 Unauthorized. Маршрути що вимагають прав адміністратора додатково

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

перевіряють роль користувача через adminMiddleware та повертають код 403 Forbidden при недостатніх правах. Паролі користувачів зберігаються виключно у хешованому вигляді та ніколи не передаються у відкритому вигляді. Система повинна коректно відображатися на пристроях з різною роздільною здатністю екрану завдяки адаптивному дизайну. Завантажувані фотографії товарів фільтруються за розширенням і дозволяються лише файли форматів jpg, jpeg, png та gif.

Окрім базових функціональних та нефункціональних вимог, до системи висуваються підвищені вимоги щодо масштабованості, безпеки та стійкості до відмов. Оскільки ринок лижного спорядження характеризується яскраво вираженою сезонністю, архітектура програмного продукту повинна підтримувати горизонтальне масштабування для безперебійної обробки пікових навантажень у зимовий період. Серверна частина повинна бути спроектована з урахуванням можливості розгортання у контейнеризованих середовищах, що дозволить оперативно виділяти додаткові обчислювальні ресурси у разі різкого збільшення кількості одночасних підключень покупців.

Ще однією важливою групою вимог є вимоги до пошукової оптимізації та логування системних подій. Завдяки використанню Next.js, клієнтська частина повинна забезпечувати серверний рендеринг або статичну генерацію сторінок каталогу, що є критично важливим для коректної індексації товарів пошуковими роботами. Метадані кожної сторінки товару мають формуватися динамічно на основі актуальної інформації з бази даних. Для забезпечення надійності експлуатації системи передбачається сувора вимога до ведення журналу подій: усі критичні збої на стороні сервера, невдалі спроби авторизації або відхилені транзакції через платіжний шлюз Stripe повинні фіксуватися з відповідними мітками часу для подальшого аналізу адміністратором системи.

Окрему увагу приділено вимогам до безпеки даних. З огляду на необхідність обробки персональних даних, система повинна відповідати сучасним стандартам захисту. Зокрема, передбачено захист від найбільш

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

розповсюджених атак, таких як ін'єкції або несанкціоноване маніпулювання запитамі, шляхом валідації вхідних даних та використання Mongoose для виключення можливості несанкціонованого доступу до бази даних. Захист від міжсайтового скриптингу реалізовано через механізми автоматичного екранування даних на стороні Next.js, а захист від підробки міжсайтових запитів досягається шляхом обмеження заголовків CORS для всіх API-запитів.

Система також повинна забезпечувати високий рівень доступності. Резервне копіювання бази даних MongoDB проводиться автоматично з періодичністю не менше однієї доби, а час відновлення системи після критичного збою не повинен перевищувати тридцяти хвилин. Інтерфейс користувача підтримує можливість інтернаціоналізації, що дозволить у майбутньому легко адаптувати сайт для міжнародних ринків, надаючи підтримку мовних локалей у форматі окремих конфігураційних файлів. Окремим блоком виділено вимоги до моніторингу продуктивності. Система автоматично вимірює час виконання основних API-запитів та передає ці метрики до моніторингового інструментарію. У разі падіння швидкодії окремих вузлів системи адміністратор отримує сповіщення у режимі реального часу, що забезпечує прозорість моніторингу всіх критичних операцій в інтернет-магазині та гарантує, що жодне замовлення не буде втрачене через технічну помилку. Такий підхід до архітектури гарантує надійність, безпеку та високу якість обслуговування клієнтів незалежно від інтенсивності навантаження на платформу.

Табл. 1.4

Системні вимоги до програмного та апаратного забезпечення

Компонент	Мінімальні вимоги	Рекомендовані вимоги
ОЗУ сервера	2 ГБ	4 ГБ і більше
Node.js	v18.0	v20.0 і вище
MongoDB	v6.0	v7.0 і вище
Браузер клієнта	Chrome 90+, Firefox 88+	Остання актуальна версія

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк. 23
Змн.	Арк.	№ докум.	Підпис	Дата		

Мережа	1 Мбіт/с	10 Мбіт/с і більше
Вільне місце на диску	500 МБ	2 ГБ і більше

Висновки до першого розділу

У першому розділі проведено комплексний аналіз предметної галузі інтернет-торгівлі лижним спорядженням. Описано склад та особливості лижного обладнання як товарної категорії, а також проаналізовано тенденції ринку електронної комерції в Україні. Встановлено, що обсяг ринку онлайн-торгівлі щороку зростає на 25–30%, а попит на лижне спорядження має виражену сезонність з піком у жовтні–березні.

Досліджено сім безпосередніх аналогів програмного продукту, серед яких: Extremstyle, Unisport, Keeperstore, Decathlon, Girski, Rozetka. На основі порівняльного аналізу виявлено, що великі маркетплейси (Rozetka, Decathlon) мають перевантажений інтерфейс та низьку спеціалізацію, тоді як профільні магазини (Extremstyle, Keeperstore, Unisport) часто мають застарілу архітектуру, низьку швидкість завантаження сторінок та обмежений функціонал для персоналізації. Зокрема, жоден з аналогів не поєднує одночасно вузьку спеціалізацію на лижному обладнанні, безпечну JWT-аутентифікацію та динамічний список бажань з функцією швидкого очищення. На основі цих недоліків визначено конкурентні переваги розроблюваної системи «SkiZone».

Обґрунтовано вибір сучасного технологічного стеку для реалізації проєкту. Обрано фреймворк Next.js 15 для фронтенду, що забезпечує серверний рендеринг (SSR) та високі показники SEO-оптимізації для інтернет-магазину. Для розробки серверної частини обрано програмну платформу Node.js 18 та фреймворк Express.js 4, які дозволяють створити швидке, асинхронне та масштабоване API. Як систему керування базами даних обрано документ-орієнтовану СУБД MongoDB 6 через її гнучкість при роботі з динамічними характеристиками товарів.

На основі проведеного аналізу сформовано конкретні вимоги до системи, серед яких функціональні (каталог із фільтрацією, авторизація користувачів, модулі кошика, списку бажань та адмін-панелі), нефункціональні (швидкість відгуку до 2 секунд, безпека даних через JWT), а також системні вимоги до програмного й апаратного забезпечення клієнтської та серверної частин.

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ЛИЖНОГО ОБЛАДНАННЯ

2.1 Визначення варіантів використання системи

Процес проєктування будь-якої сучасної веборієнтованої системи починається з детального аналізу вимог користувачів та визначення функціональних меж програмного продукту [30]. Для формалізації цих вимог у сучасній інженерії програмного забезпечення використовується методологія UML (Unified Modeling Language), зокрема діаграми варіантів використання (Use Case Diagrams) та діаграми класів (Class Diagrams) [20]. Такий підхід дозволяє відокремити зовнішню поведінку системи від її внутрішньої архітектурної реалізації, забезпечуючи чітке розуміння взаємодії користувачів із бізнес-логікою додатка.

Табл. 2.1

Опис варіантів використання системи SkiZone

Назва UC	Актор	Короткий опис
Переглянути каталог	Гість, Користувач	Перегляд товарів з пошуком, фільтрацією та сортуванням
Зареєструватись	Гість	Введення логіну та паролю, хешування bcrypt
Авторизуватись	Гість	Вхід до системи, отримання JWT-токена терміном 1 доба
Додати до кошика	Користувач	Додавання товару; повторне — збільшення quantity на 1
Оформити замовлення	Користувач	Вибір доставки та способу оплати (cash/card)
Керувати списком бажань	Користувач	Додавання, перегляд, очищення вішліста
Редагувати профіль	Користувач	Оновлення логіну та паролю, перегляд замовлень
Керувати товарами	Адміністратор	CRUD-операції + завантаження фото (jpg/jpeg/png/gif)

Керувати замовленнями	Адміністратор	Перегляд усіх замовлень, зміна статусу
-----------------------	---------------	--

Для побудови діаграми варіантів використання (рис. 2.1) використано PlantUML. Для генерації зображення необхідно перейти на сайт <https://www.planttext.com/>, вставити відповідний код та зберегти результат у форматі PNG. Діаграма варіантів використання системи SkiZone охоплює трьох акторів: Гість, Користувач та Адміністратор. Гість є незареєстрованим відвідувачем платформи, якому доступні лише базові функції — перегляд каталогу товарів, реєстрація нового облікового запису та авторизація в системі. Після успішної авторизації Гість отримує статус Користувача, що розширює його можливості взаємодії з платформою. Авторизований Користувач може додавати товари до кошика, оформлювати замовлення з вибором способу оплати, керувати списком бажань — додавати товари, переглядати їх та очищати список, а також редагувати власний профіль та переглядати історію замовлень. Адміністратор є привілейованим актором системи, який окрім усіх можливостей звичайного користувача має доступ до керування каталогом товарів через повний цикл CRUD-операцій із завантаженням фотографій, а також до перегляду всіх замовлень від усіх користувачів із можливістю зміни їх поточного статусу.

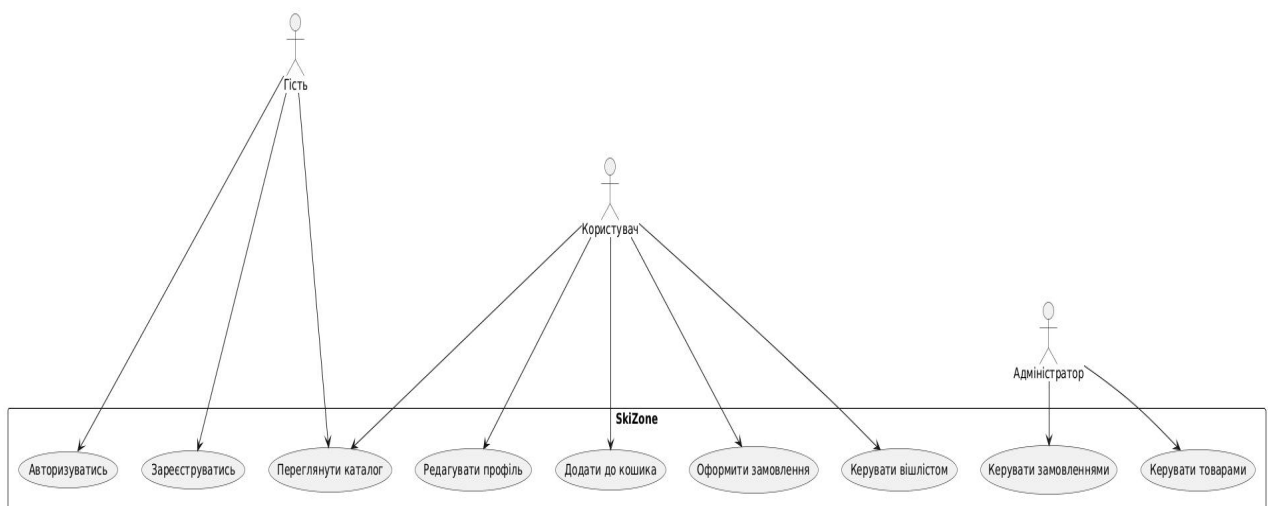


Рис. 2.1 Діаграма варіантів використання системи SkiZone

Діаграма потоків даних відображає повний цикл обробки інформації в системі SkiZone від ініціювання запиту користувачем до отримання відповіді. На верхньому рівні діаграми розміщено актора User, який взаємодіє з системою через три основні операції: додавання товару до кошика, оформлення замовлення та перегляд каталогу продукції. Усі ці дії передаються до компонента Frontend, реалізованого на базі Next.js, який виступає проміжною ланкою між користувачем та серверною частиною. Frontend формує відповідні API-запити та передає їх до компонента Backend, реалізованого на Node.js з Express.js. Бекенд у свою чергу виконує чотири ключові операції: отримання списку товарів з бази даних через Fetch Products, оновлення стану кошика через Update Cart API, створення нового замовлення через Create Order та відображення повідомлення про успіх через Show Success. Для збереження та отримання даних Backend звертається до бази даних MongoDB, виконуючи операції Update Cart in DB, Save Order та Query DB. Під час оформлення замовлення Frontend передає дані до Backend, який створює новий запис замовлення в базі даних MongoDB, після чого очищує кошик користувача та повертає підтвердження про успішне виконання операції. Отримане підтвердження використовується Frontend для відображення повідомлення про успішне оформлення замовлення. Результати всіх операцій повертаються через Frontend до користувача у вигляді оновленого інтерфейсу.

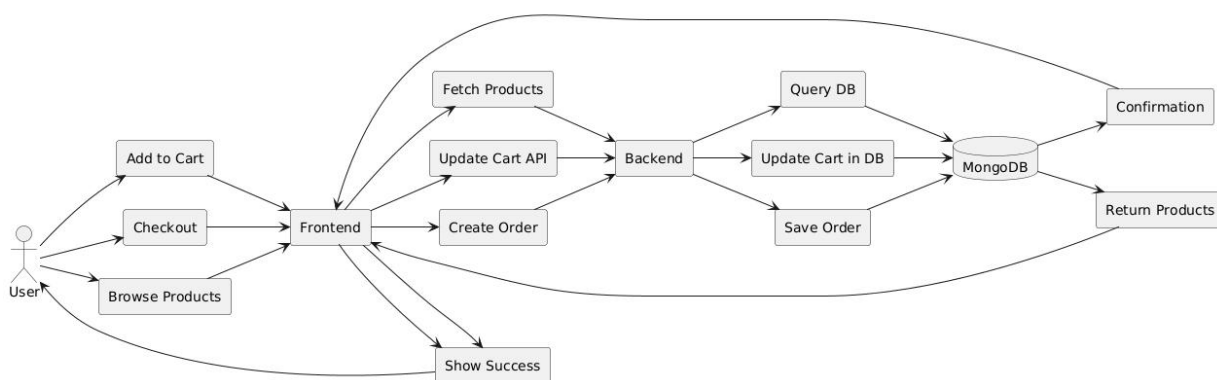


Рис. 2.2 - Діаграма потоків даних: повний цикл операцій системи

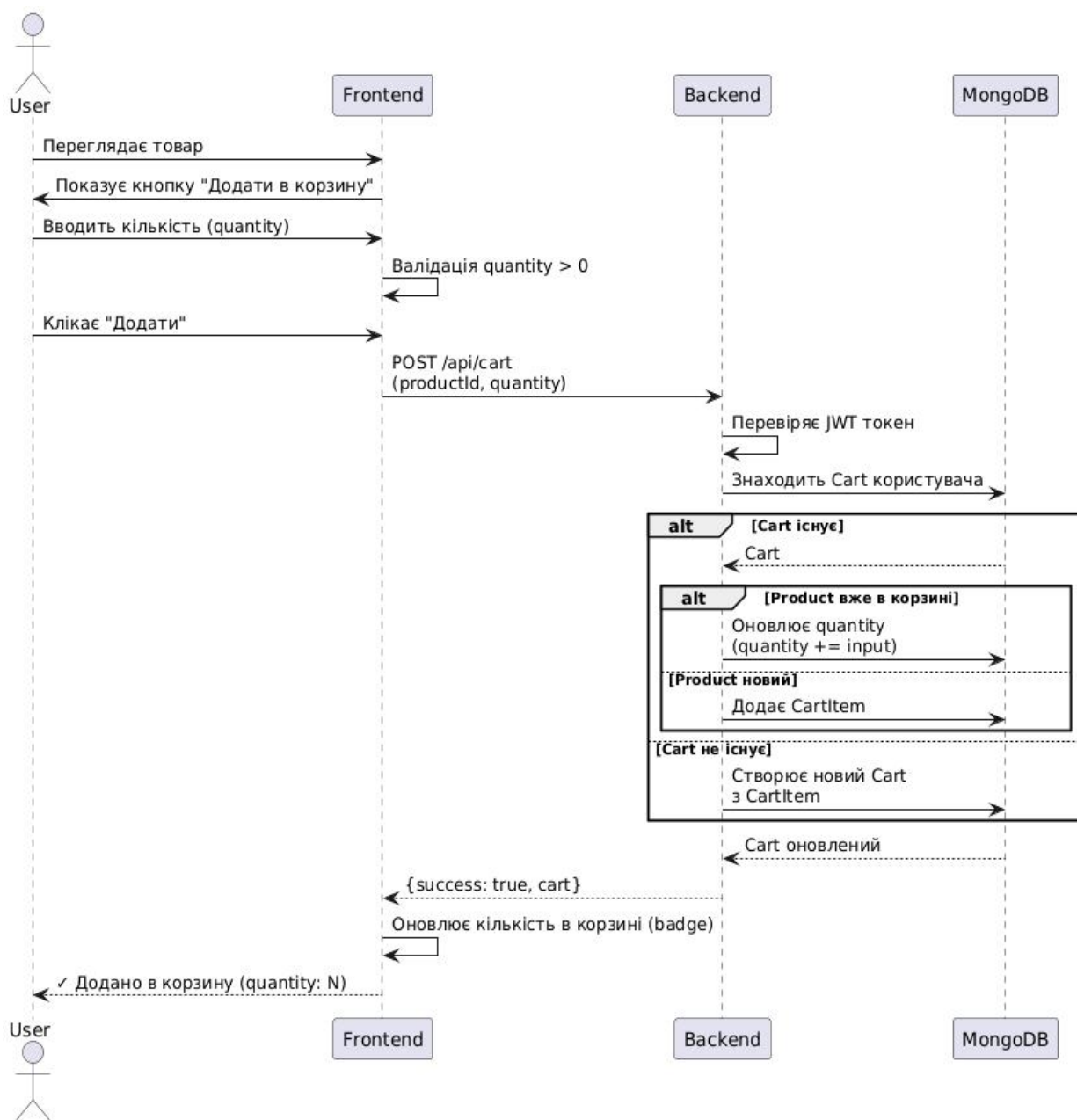


Рис. 2.3 - Діаграма послідовності: додавання товару до кошика покупок

Діаграма послідовності процесу додавання товару до кошика відображає взаємодію чотирьох учасників: User, Frontend, Backend та MongoDB. Процес починається з того, що користувач переглядає сторінку товару, де Frontend відображає кнопку «Додати в кошик». Користувач вводить потрібну кількість товару, після чого Frontend виконує валідацію введеного значення — перевіряє що quantity більше нуля. Після успішної валідації користувач натискає кнопку «Додати», і Frontend надсилає POST-запит на маршрут /api/cart з параметрами productId та quantity. Backend отримує запит та першочергово перевіряє наявність та валідність JWT-токена

в заголовку Authorization. Після підтвердження автентичності користувача Backend звертається до MongoDB для пошуку існуючого документа Cart за ідентифікатором користувача. Якщо документ кошика існує в базі даних, система перевіряє чи присутній даний товар у масиві items. У разі якщо товар вже є в кошику, система збільшує значення поля quantity на введену користувачем кількість. Якщо товар додається вперше, система додає новий елемент CartItem до масиву. У випадку коли документ Cart для цього користувача ще не існує, Backend створює новий документ із першим CartItem. Після збереження оновленого стану кошика в MongoDB Backend повертає Frontend об'єкт у форматі success: true, cart, після чого Frontend оновлює відображення кількості товарів у кошику у навігаційній панелі та повідомляє користувача про успішне додавання.

Діаграма послідовності керування списком бажань охоплює повний цикл взаємодії користувача з вішлістом через чотири компоненти: User, Frontend, Backend та MongoDB. Процес додавання товару до списку бажань починається з того, що користувач переглядає картку товару та натискає кнопку «Додати в бажання». Frontend надсилає POST-запит на маршрут /api/wishlist з параметром productId. Backend знаходить документ Wishlist поточного користувача в MongoDB. Якщо документ Wishlist вже існує, система додає ідентифікатор товару до масиву products та зберігає оновлений документ. Якщо документ Wishlist ще не існує, Backend створює новий документ із першим товаром. Після успішного збереження Frontend отримує підтвердження success: true та відображає користувачу повідомлення про додавання товару до списку бажань. Для перегляду списку бажань користувач переходить на відповідну сторінку, Frontend надсилає GET-запит на /api/wishlist, Backend виконує запит до MongoDB з автоматичним заповненням деталей товарів через метод populate та повертає повний список товарів з усіма характеристиками. При видаленні окремого товару Frontend надсилає DELETE-запит на /api/wishlist/:productId, Backend видаляє

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

відповідний ідентифікатор з масиву products та зберігає оновлений документ, після чого Frontend оновлює відображення списку.

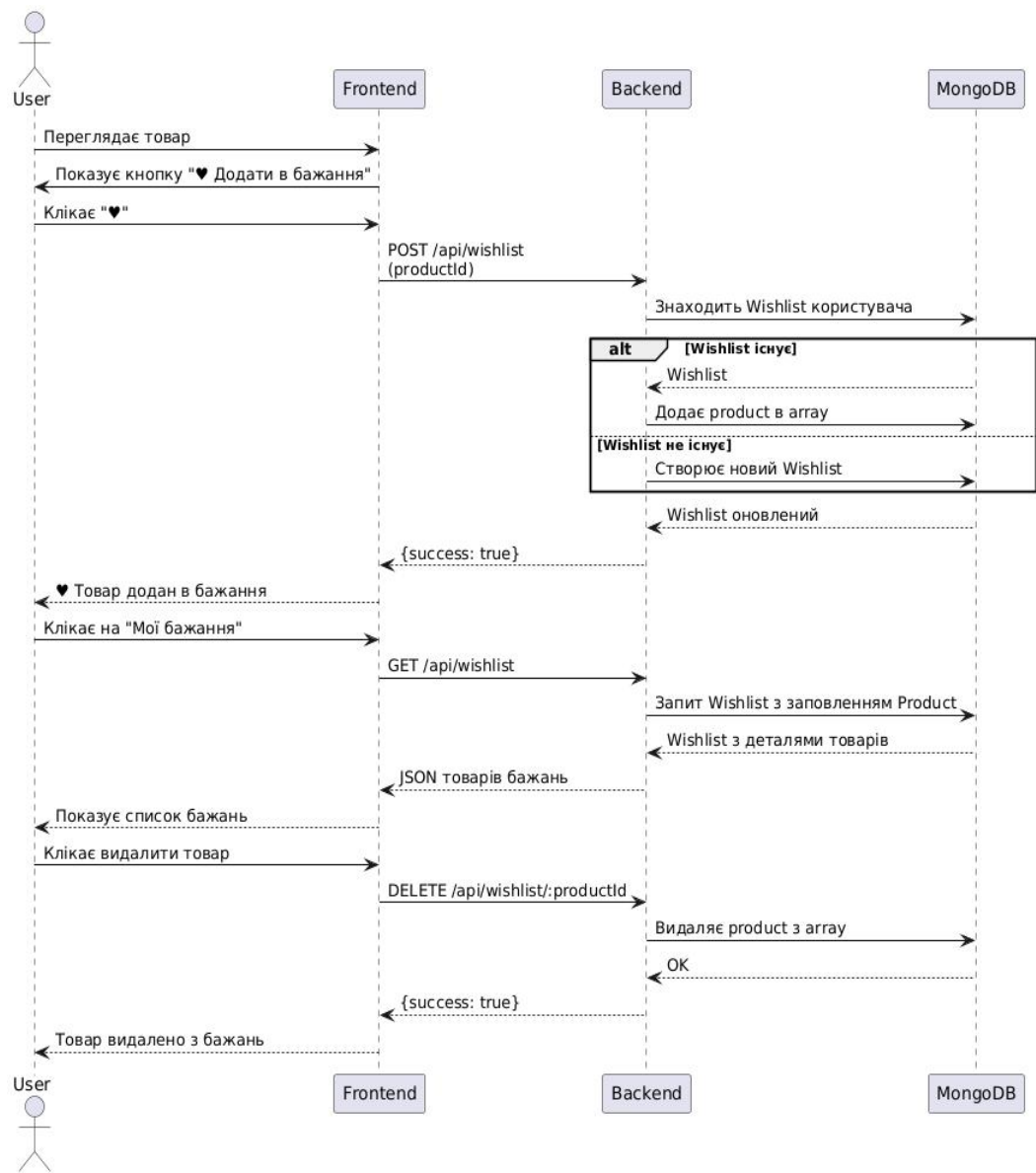


Рис.2.4 - Діаграма послідовності: керування списком бажань користувача.

2.2 Розробка бази даних системи

Для забезпечення високої масштабованості, швидкодії та гнучкості інформаційної структури e-commerce платформи SkiZone було спроектовано архітектуру збереження даних на базі документо-орієнтованої СУБД MongoDB у поєднанні з серверною логікою на Express.js. На відміну від класичних реляційних моделей, обраний підхід дозволяє зберігати інформацію у вигляді динамічних JSON-подібних документів [21]. Це значно

спрощує структурування вкладених сутностей, таких як елементи кошика або позиції замовлення, усуваючи необхідність виконання високонавантажених операцій об'єднання таблиць (JOIN) на рівні бази даних [22]. Повну архітектурну схему взаємодії компонентів безпеки, контролерів бізнес-логіки та моделей даних представлено на рисунку 2.5.

На основі розробленої діаграми чітко прослідковується взаємодія між моделями даних та Express-контролерами, де особлива увага приділена забезпеченню транзакційності та цілісності інформації під час виконання фінансових операцій. При ініціалізації користувачем процесу оформлення покупки, функціональний модуль `orderController` здійснює транзакційне зчитування поточного стану кошика з колекції `Cart` для конкретного ідентифікатора користувача. Після цього сервер виконує агрегацію даних за допомогою методу `.populate()`, зіставляючи додані товари з актуальними цінами та характеристиками з колекції `Product`. На основі отриманих даних обчислюється підсумкова сума замовлення `totalAmount`, створюється новий документ у колекції `Order` із первинним статусом обробки, і лише за умови успішного виконання всіх попередніх кроків бекенд повністю очищує масив елементів у кошику користувача. Така послідовність операцій повністю виключає логічні помилки розсинхронізації даних у системі.

Додатковою перевагою розробленої структури бази даних є логічний розподіл інформації між окремими колекціями. Такий підхід забезпечує чітке розмежування відповідальності між різними сутностями системи, спрощує обробку даних та полегшує їх подальше адміністрування. Колекція `User` використовується для зберігання облікових записів користувачів, `Product` містить інформацію про товари каталогу, `Cart` забезпечує збереження товарів, доданих до кошика, а `Order` використовується для фіксації оформлених замовлень. Для реалізації функції списку бажань передбачено окрему колекцію `Wishlist`, що дозволяє користувачам зберігати товари для подальшого перегляду та швидко повертатися до них під час наступних відвідувань вебдодатку.

Взаємодія між зазначеними колекціями здійснюється за допомогою ідентифікаторів документів та можливостей бібліотеки Mongoose. Для отримання повної інформації про пов'язані об'єкти використовується метод `populate()`, який автоматично підвантажує необхідні документи з інших колекцій. Завдяки цьому зменшується дублювання даних, забезпечується їх актуальність та спрощується реалізація серверної логіки. Крім того, використання посилань між документами дозволяє підтримувати єдине джерело достовірної інформації про товари та користувачів, що позитивно впливає на цілісність даних у системі.

Запропонована структура бази даних є достатньо гнучкою для подальшого розвитку програмного продукту. За необхідності вона може бути розширена новими колекціями або полями без суттєвої зміни вже реалізованої архітектури, що є однією з переваг використання документоорієнтованої СУБД MongoDB. Наприклад, у майбутньому до системи можуть бути додані модулі відгуків користувачів, рейтингу товарів, онлайн-оплати або персональних рекомендацій без необхідності повного перепроектування структури бази даних.

Важливим архітектурним рішенням, винесеним на діаграму, є ізольована підсистема безпеки, яка контролює потік вхідних клієнтських запитів. Процес захисту персональних даних та розмежування прав доступу базується на використанні криптографічного хешування паролів алгоритмом `bcrypt` із рівнем солі `rounds: 10` на етапі реєстрації через `UserController`. Подальша автентифікація сесій реалізована за допомогою механізму JSON Web Tokens (JWT). Згенерований сервером токен передається клієнтом у HTTP-заголовку `Authorization` і перехоплюється проміжним програмним забезпеченням `authMiddleware`. Модуль безпеки виконує дешифрування та валідацію підпису токена, імпортуючи ідентифікатор користувача та його роль безпосередньо в об'єкт запиту. Рольова модель доступу за допомогою методу `checkAdminRole()` забезпечує суворе блокування спроб несанкціонованої модифікації каталогу товарів, обмежуючи доступ до

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

методів створення, редагування та видалення в productController виключно для користувачів із привілеями адміністратора.

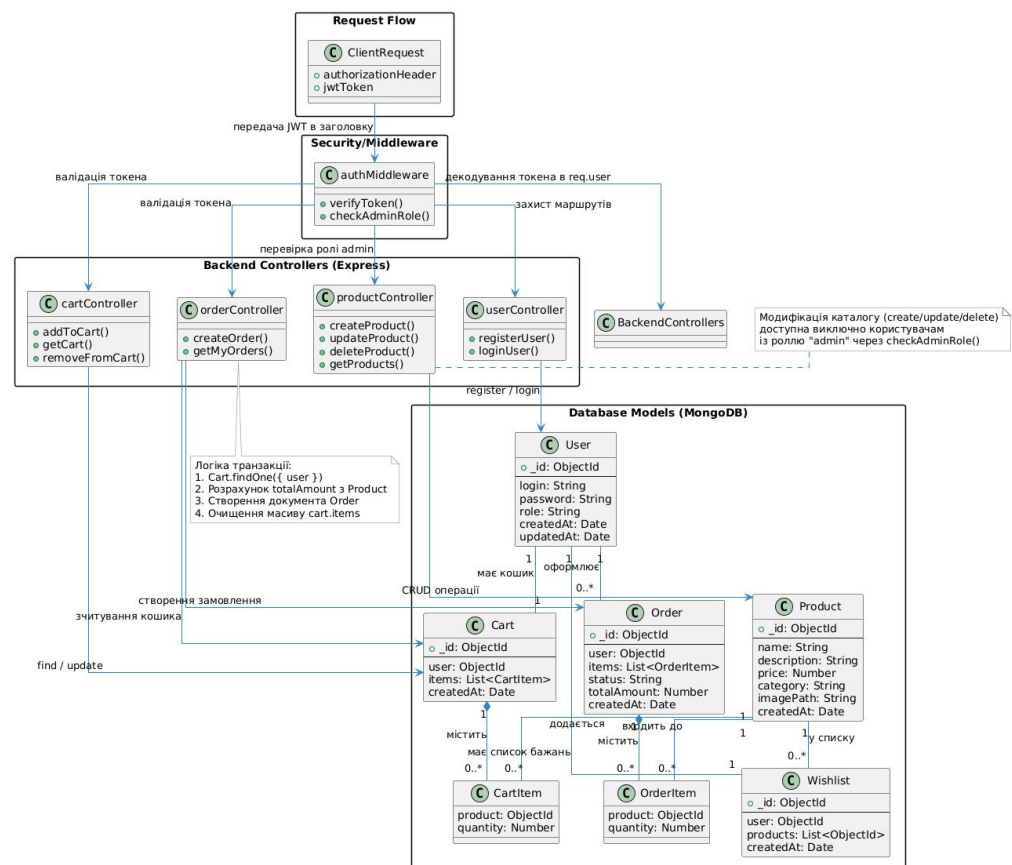


Рис.2.5 - Діаграма бази даних

2.3 Проектування алгоритмів роботи системи

Діаграма активності процесу аутентифікації користувача відображає повний алгоритм як реєстрації так і авторизації в системі SkiZone. Процес починається з введення користувачем логіну та паролю у відповідну форму, після чого Frontend надсилає POST-запит на маршрут /api/users/login. На першому кроці обробки Backend виконує пошук користувача в колекції users за полем login. Якщо користувача з таким логіном не знайдено в базі даних, система повертає HTTP-статус 401 Unauthorized та відображає модальне вікно з повідомленням «Невірний логін або пароль». Якщо користувача знайдено, алгоритм переходить до наступного кроку — порівняння введеного паролю з хешованим значенням збереженим у базі даних через функцію

bcrypt.compare. У разі невідповідності паролів система також повертає статус 401 та відповідне повідомлення про помилку. При успішному збігу паролів Backend генерує JWT-токен за допомогою функції jwt.sign з payload що містить userId та role користувача, підписує його секретним ключем JWT_SECRET та встановлює термін дії рівний одній добі через параметр expiresIn зі значенням 1d. Згенерований токен разом з роллю користувача повертається клієнту, Frontend зберігає їх у localStorage браузера та перенаправляє користувача на головну сторінку магазину.

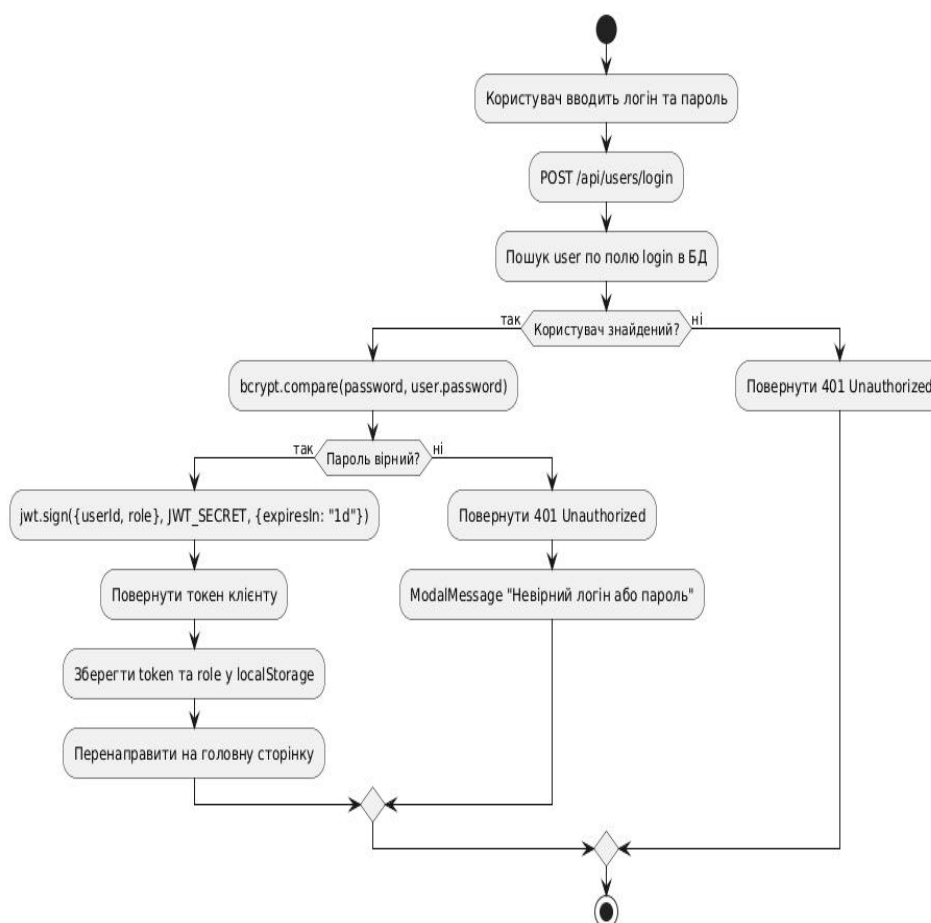


Рис. 2.6 – Діаграма активності: алгоритм аутентифікації користувача

Діаграма активності алгоритму розрахунку загальної суми кошика відображає покроковий процес обчислення вартості всіх товарів обраних користувачем. На початку алгоритму система отримує список товарів кошика з MongoDB через GET-запит на /api/cart з використанням методу populate для отримання повних даних про кожен товар включаючи його ціну. Після отримання масиву items змінна-акумулятор totalSum ініціалізується

значенням нуль. Далі алгоритм входить у цикл ітерації по кожному елементу масиву `items`. На кожній ітерації система перевіряє наявність наступного елемента в масиві. Якщо елемент існує, алгоритм отримує поточний елемент `item` та обчислює його вартість шляхом множення ціни товару `item.product.price` на кількість `item.quantity`, результат записується у змінну `subtotal`. Отримане значення `subtotal` додається до акумулятора `totalSum`. Цикл продовжується до тих пір поки не буде оброблено всі елементи масиву `items`. Після завершення циклу алгоритм виводить фінальне значення `totalSum` як загальну суму кошика, яка відображається користувачу на сторінці кошика та у формі оформлення замовлення.

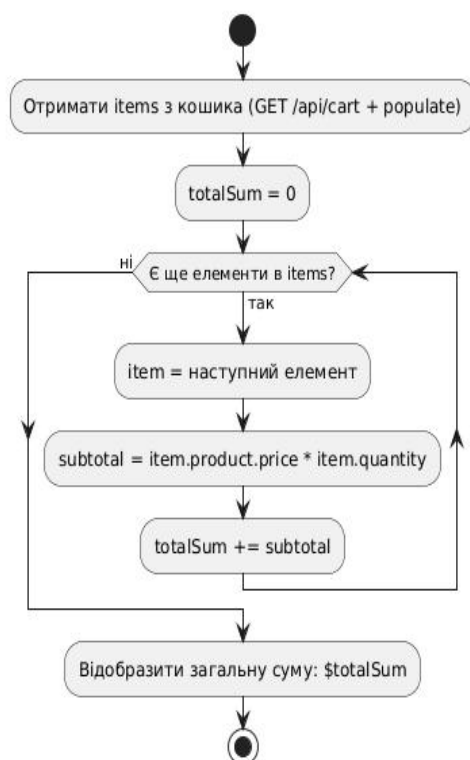


Рис. 2.7 – Діаграма активності: алгоритм розрахунку суми кошика

Діаграма активності алгоритму фільтрації товарів відображає логіку формування динамічного запиту до бази даних на основі параметрів введених користувачем. Процес починається з отримання від користувача двох параметрів — пошукового запиту у текстовому полі та вибраної категорії зі списку. На першому кроці Backend ініціалізує порожній об'єкт фільтра `queryFilter`. Далі алгоритм перевіряє наявність параметра `category` у

запиті. Якщо категорію обрано, до фільтра додається умова `filter.category` зі значенням обраної категорії. Наступним кроком алгоритм перевіряє наявність пошукового запиту `search`. Якщо пошуковий рядок не порожній, система створює об'єкт регулярного виразу `RegExp` з прапором нечутливості до регістру та формує складену умову `filter.$or` що включає два поля — `name` та `description` — кожне з умовою `$regex`. Така реалізація дозволяє знаходити товари при частковому збігу пошукового запиту як з назвою так і з описом товару одночасно. Після формування повного об'єкта фільтра Backend виконує запит до MongoDB через метод `Product.find` з передачею сформованого `queryFilter`. База даних повертає масив товарів що задовольняють усім умовам фільтрації, після чого Backend відправляє результат клієнту зі статусом 200, а Frontend відображає відфільтрований список товарів на сторінці каталогу.

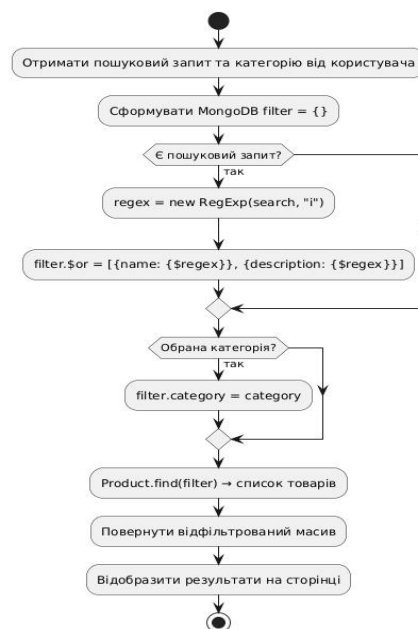


Рис. 2.8 – Діаграма активності: алгоритм фільтрації товарів

Діаграма активності управління каталогом товарів відображає три основні операції доступні адміністратору системи: додавання нового товару, редагування існуючого та видалення застарілого. Процес починається з авторизації адміністратора в системі — якщо авторизація невдала, алгоритм завершується з повідомленням про помилку. Після успішної авторизації адміністратор переглядає каталог товарів та обирає одну з трьох дій. У разі

вибору операції додавання товару адміністратор заповнює форму що містить поля назви, опису, ціни та категорії, після чого завантажує зображення товару. Система валідує введені дані та зберігає новий документ у колекції products MongoDB разом із шляхом до завантаженого файлу [27]. При успішному збереженні відображається повідомлення про успіх, при помилці — відповідне повідомлення про невдачу. У разі вибору операції редагування адміністратор обирає товар зі списку та змінює потрібні поля у формі редагування, після чого система оновлює документ у MongoDB через метод findByIdAndUpdate. При виборі операції видалення адміністратор обирає товар та підтверджує видалення, після чого Backend виконує запит deleteOne до MongoDB та видаляє відповідний документ з колекції. Після завершення будь-якої операції адміністратор може продовжити роботу з каталогом або завершити сеанс.

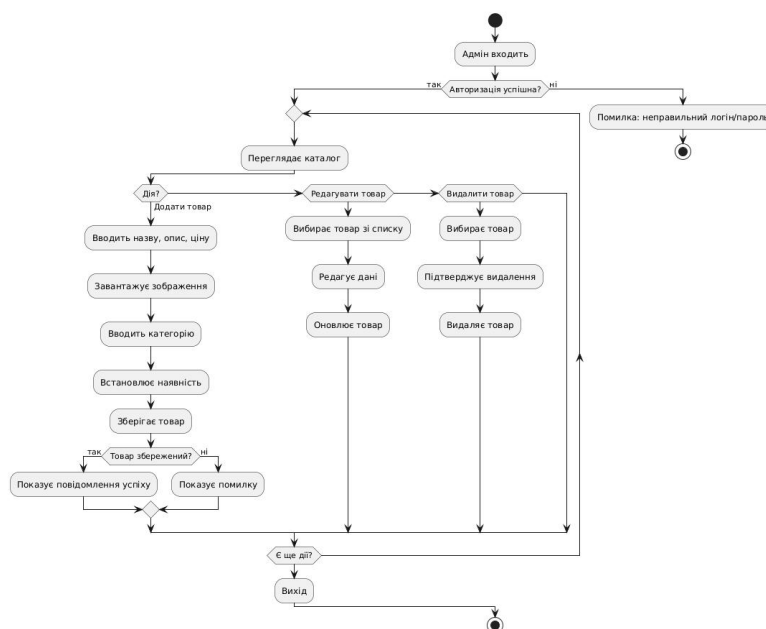


Рис. 2.9 - Діаграма активності: управління каталогом товарів.

Процес оформлення замовлення є одним із ключових бізнес-процесів системи SkiZone. Після заповнення користувачем необхідних даних для доставки клієнтська частина надсилає запит до серверної частини. Сервер виконує перевірку автентифікації користувача за JWT-токеном, отримує актуальний вміст кошика з бази даних, обчислює загальну суму замовлення та створює новий документ у колекції Order зі статусом Processing. Після

успішного створення замовлення кошик користувача очищується, а клієнт отримує повідомлення про успішне оформлення замовлення.



Рис. 2.10 - Діаграма активності: процес оформлення замовлення.

Змн.	Арк.	№ докум.	Підпис	Дата

Діаграма послідовності оформлення замовлення відображає взаємодію чотирьох основних учасників: User, Frontend, Backend та MongoDB. Процес починається з натискання користувачем кнопки «Оформити замовлення», після чого Frontend виконує локальну перевірку введених даних та надсилає POST-запит до маршруту /api/orders/create. Backend після успішної автентифікації користувача за JWT-токеном отримує актуальний вміст кошика з бази даних MongoDB, обчислює загальну вартість замовлення та створює новий документ у колекції Order зі статусом Processing. Після успішного збереження замовлення сервер очищує кошик користувача, зберігає зміни в базі даних та повертає відповідь про успішне оформлення замовлення. Отримавши відповідь сервера, Frontend відображає повідомлення про успішне створення замовлення та очищує локальний стан кошика.

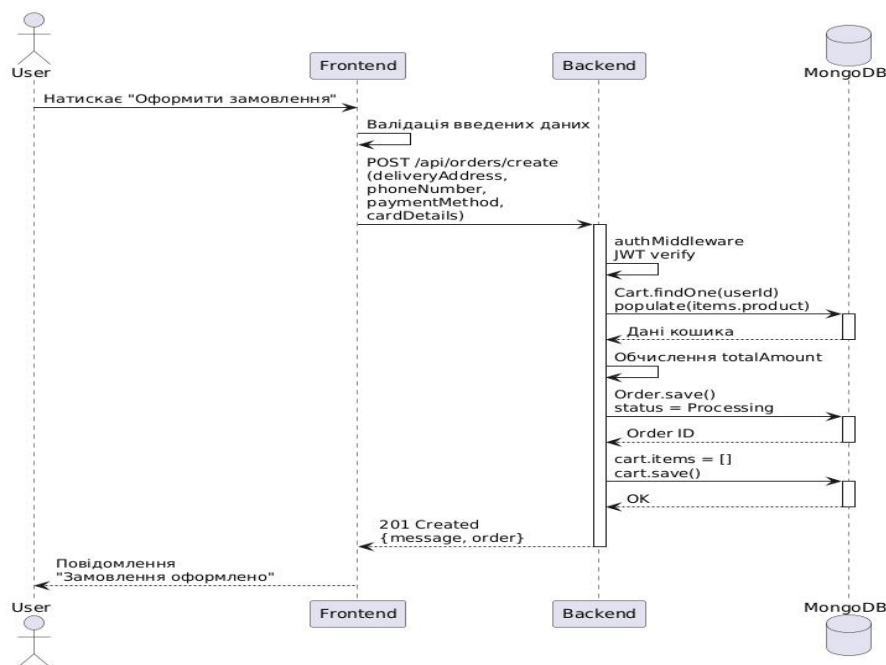


Рис.2.11 - Діаграма послідовності: оформлення замовлення.

Діаграму розгортання системи (рис. 2.6), що відображає фізичну архітектуру та взаємодію між компонентами, рекомендується побудувати за наступним PlantUML

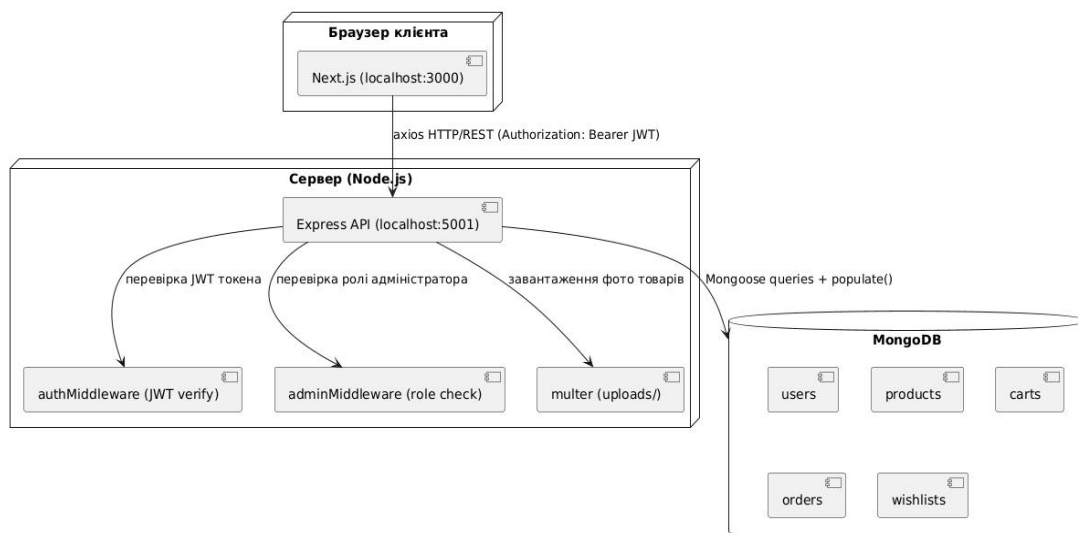


Рис. 2.12 – Діаграма розгортання системи SkiZone

2.4 Реалізація серверної частини системи

Серверна частина організована за архітектурним шаблоном MVC (Model-View-Controller): моделі Mongoose у `models/`, логіка обробки запитів у `controllers/`, API-маршрути у `routes/`, middleware у `middleware/` [23]. Така організація забезпечує чітке розподілення відповідальності та полегшує подальшу підтримку і масштабування.

Модульний підхід полягає у розділенні функціональності на незалежні компоненти, що взаємодіють через RESTful API. Архітектура включає такі модулі: управління користувачами (`UserRoutes` → `UserController` → `User`), управління товарами (`ProductRoutes` → `ProductController` → `Product`), управління кошиком (`CartRoutes` → `CartController` → `Cart`), управління замовленнями (`OrderRoutes` → `OrderController` → `Order`), управління вішлістом (`WishlistRoutes` → `WishlistController` → `Wishlist`).

Забезпечення безпеки реалізовано шляхом використання middleware-компонента `authMiddleware.js`. Він виконує перевірку автентичності користувача у три етапи: витягує JWT-токен із заголовка `Authorization` (формат `Bearer <token>`), верифікує його за допомогою функції `jwt.verify(token, JWT_SECRET)` та, у разі успішної перевірки, зберігає інформацію про користувача у властивості `req.user`, після чого передає

керування наступному обробнику запиту. Якщо токен відсутній, недійсний або прострочений, сервер повертає відповідь з кодом 401 Unauthorized.

Контроль прав адміністратора реалізовано безпосередньо в контролерах системи. Перед виконанням операцій створення, редагування або видалення товарів здійснюється перевірка ролі користувача (req.user.role === "admin"). Якщо користувач не має прав адміністратора, сервер повертає відповідь з кодом 403 Forbidden.

Табл.2.5

API-маршрути системи SkiZone

Метод	Маршрут	Захист	Опис
POST	/api/users/register	–	Реєстрація, bcrypt.hash(pwd, 10)
POST	/api/users/login	–	Авторизація, повертає JWT (expiresIn: 1d)
GET	/api/users/profile	auth	Профіль поточного користувача
PATCH	/api/users/update	auth	Оновлення логіну або паролю
GET	/api/products	–	Список товарів (з фільтром RegExp)
GET	/api/products/:id	–	Товар за ідентифікатором
POST	/api/products	auth + upload	Додати товар (адмін)
PUT	/api/products/:id	auth + upload	Редагувати товар (адмін)
DELETE	/api/products/:id	auth	Видалити товар (адмін)
GET	/api/cart	auth	Кошик з populate(items.product)
POST	/api/cart/add	auth	Додати або збільшити quantity
DELETE	/api/cart/remove/:id	auth	Видалити товар з кошика
GET	/api/orders/my	auth	Замовлення поточного користувача
POST	/api/orders/create	auth	Оформити замовлення
GET	/api/wishlist	auth	Список бажань
POST	/api/wishlist/add	auth	Додати до вішліста (унікальний індекс)

POST	/api/wishlist/add	auth	Додати до вішліста (унікальний індекс)
DELETE	/api/wishlist/remove/:productId	auth	Видалити з вішліста
DELETE	/api/wishlist/clear	auth	Очистити весь вішліст

Розглянемо детально логіку роботи кожного контролера серверної частини. Контролер користувачів `UserController.js` реалізує чотири функції управління обліковими записами. Функція `registerUser` зчитує логін та пароль з тіла запиту і перевіряє унікальність логіну в базі даних — у разі збігу повертає відповідь з кодом 400. При відсутності дублікату пароль хешується через `bcrypt.hash(password, 10)` та зберігається разом з логіном у колекції `users`, після чого повертається код 201. Функція `loginUser` виконує автентифікацію користувача шляхом пошуку облікового запису за логіном та перевірки відповідності введеного пароля збереженому хешу за допомогою функції `bcrypt.compare()`. У разі успішної перевірки формується JWT-токен, який містить ідентифікатор користувача та його роль, після чого повертається клієнту разом із кодом успішного виконання. Якщо користувача не знайдено або пароль введено неправильно, сервер повертає повідомлення про помилку з кодом 400

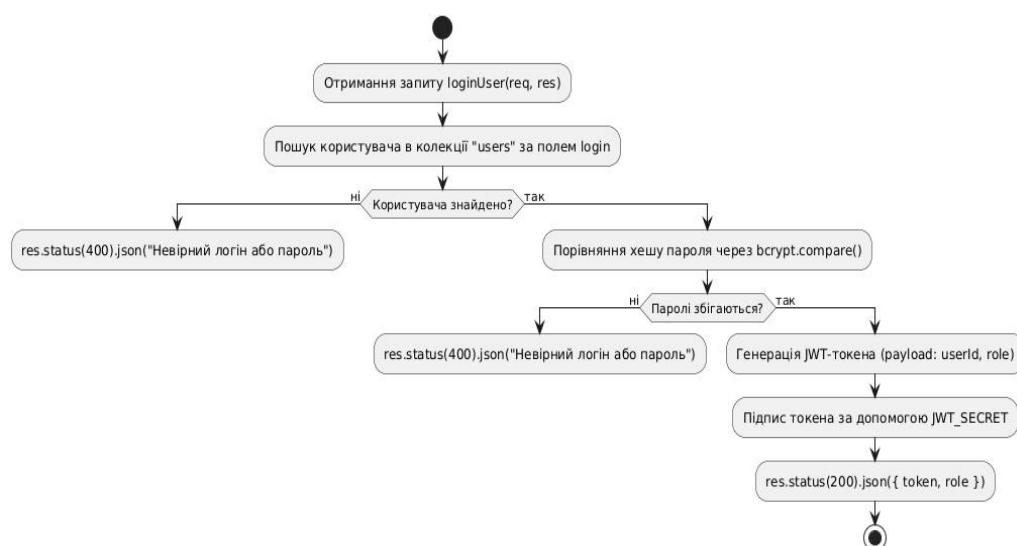


Рис. 2.13 – Діаграма контролер користувачів

Функція `loginUser` знаходить користувача за полем `login`, перевіряє пароль через `bcrypt.compare` та у разі успіху генерує JWT-токен з `payload { userId: user._id, role: user.role }`, підписує його секретним ключем `JWT_SECRET` та встановлює термін дії 1 доба, повертаючи токен та роль клієнту. Функція `getProfile` повертає логін поточного користувача за ідентифікатором з `req.user.userId` після проходження `authMiddleware`. Функція `updateUser` дозволяє авторизованому користувачу оновити логін або пароль — новий пароль перед збереженням повторно хешується через `bcrypt`.

Контролер товарів `productController.js` реалізує повний набір CRUD-операцій. Функція `getAllProducts` зчитує параметри `search` та `category` з рядка запиту і формує MongoDB-фільтр: якщо передано пошуковий запит, він застосовується до полів `name` та `description` через регулярний вираз нечутливий до регістру — це дозволяє знаходити товари при частковому збігу назви або опису.



Рис. 2.14 – Діаграма контроллер товарів

Функція `createProduct` перетворює ціну на число, перевіряє що вона не від'ємна та зберігає шлях до завантаженого файлу зображення з `req.file.path`, що формується middleware `Multer`. Функції `updateProduct` та `deleteProduct` доступні лише користувачам з роллю `admin` — `updateProduct` використовує `findByIdAndUpdate` з опцією `{ new: true }` для повернення оновленого документа.

Контролер кошика `cartController.js` реалізує логіку управління вмістом кошика. Функція `getCart` знаходить кошик поточного користувача та виконує `populate('items.product')` для отримання повних об'єктів товарів замість `ObjectId`-посилань. Якщо кошика не існує — повертається порожній масив `{ items: [] }`. Функція `addToCart` реалізує розумну логіку: якщо кошика не існує — створює новий документ `Cart`, якщо товар вже є в масиві `items` — збільшує його `quantity` на передане значення, інакше додає новий елемент `{ product: productId, quantity }`. Функція `removeFromCart` фільтрує масив `items` видаляючи елементи з відповідним `productId` та зберігає оновлений кошик.

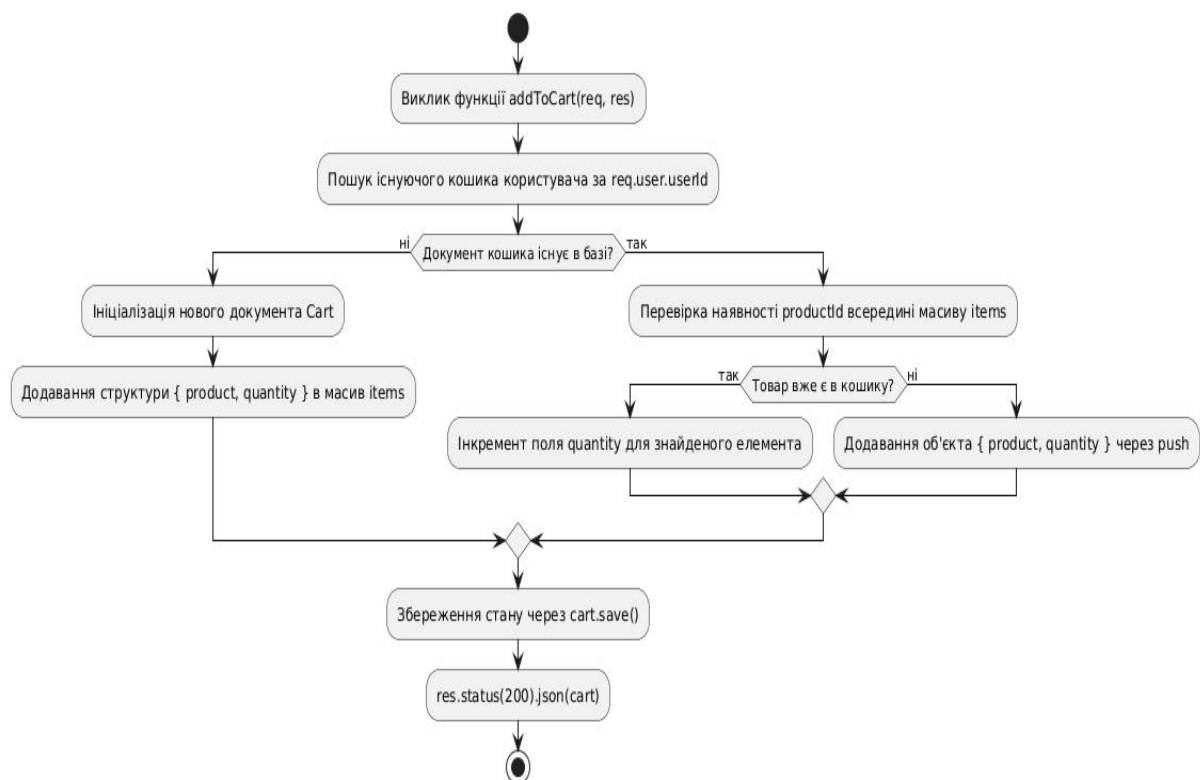


Рис. 2.15 – Діаграма контролер кошика

Контролер замовлень `orderController.js` містить дві функції. Функція `getMyOrders` повертає замовлення поточного користувача відсортовані за

датою у спадному порядку з `populate('items.product')`. Функція `createOrder` знаходить кошик користувача, перевіряє що він не порожній, обчислює `totalAmount` як суму добутків `item.product.price` та `item.quantity` для кожного елемента, створює документ `Order` зі статусом "Processing" та після збереження очищає кошик — це гарантує що товари не будуть замовлені двічі.

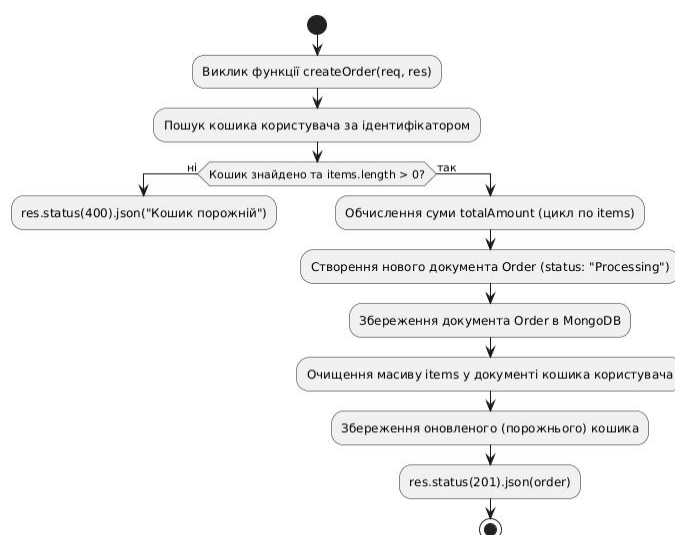


Рис. 2.16 – Діаграма контроллер замовлень

Контролер списку бажань `wishlistController.js` реалізує чотири функції. Функція `getWishlist` повертає список бажань або створює новий порожній якщо він відсутній.

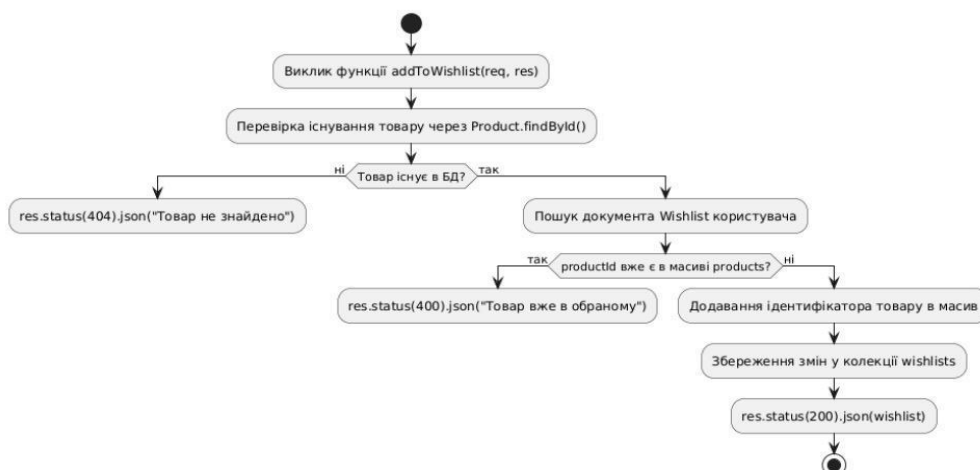


Рис. 2.17 – Діаграма контроллер замовлень

Функція `addToWishlist` спочатку перевіряє існування товару в колекції `products` через `Product.findById`, потім перевіряє що товар ще не доданий до вішліста і лише тоді додає його — це запобігає дублюванню. Унікальний індекс `{ user: 1, products: 1 }` на рівні MongoDB забезпечує додатковий захист від дублікатів. Функція `clearWishlist` очищає весь масив `products` одним запитом без перебору елементів.

Структура клієнтської частини організована за принципом розділення відповідальності між сторінками та компонентами представлення [24]. Директорія `src/pages` містить вісім сторінок маршрутизації: `index.js` підключає `HomePage`, `admin.js` підключає `AdminPage`, `login.js` підключає `LoginPage`, `register.js` підключає `RegisterPage`, `cart.js` підключає `CartPage`, `wishlist.js` підключає `WishlistPage`, `account.js` підключає `AccountPage`, а `product/[id].js` підключає `ProductPage` з динамічним параметром ідентифікатора товару. Директорія `src/views` містить відповідні компоненти представлення для кожної сторінки.

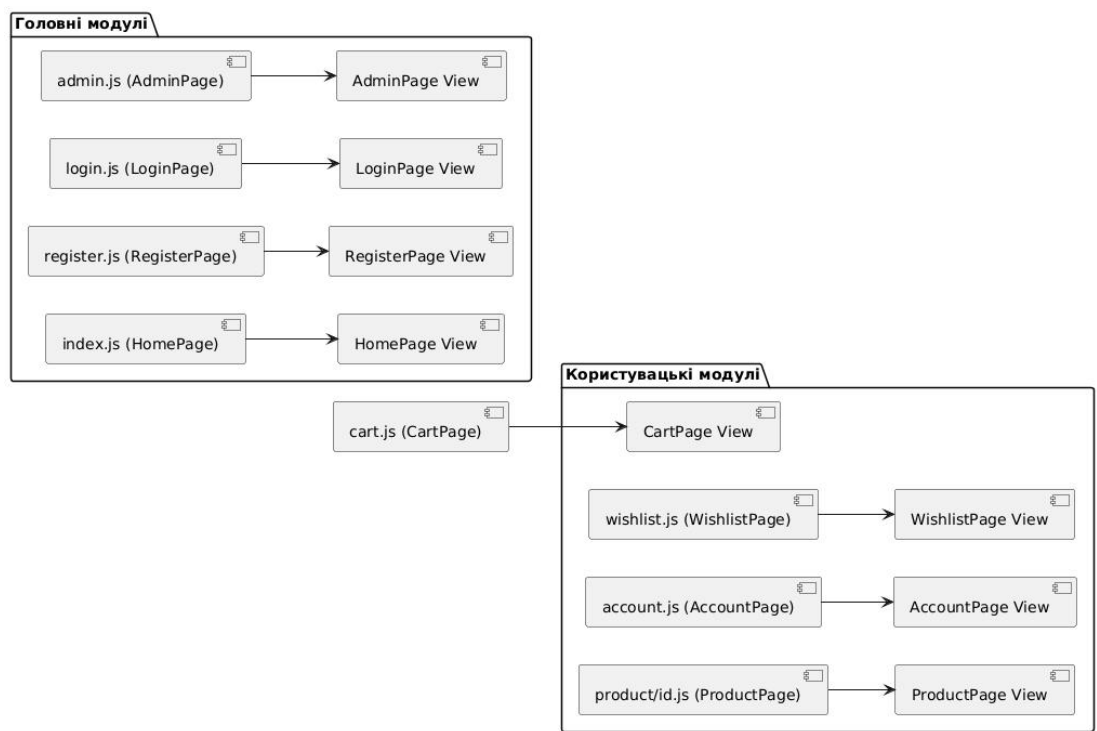


Рис. 2.18 – Діаграма структури сторінок та маршрутизації клієнської частини

Компонент `Layout.js` обгортає всі сторінки додатку, вставляє `NavBar` зверху та рендерить дочірні компоненти у тег `main`. Компонент `NavBar.jsx`

визначає стан авторизації користувача шляхом зчитування token та role з localStorage у хуку useEffect та динамічно відображає різні посилання залежно від ролі: неавторизованим користувачам показуються посилання "Увійти" та "Реєстрація", звичайним користувачам — "Акаунт", "Обране" та "Кошик", адміністратору — "Адмін-панель", "Обране" та "Кошик". Кнопка "Вийти" очищає token і role з localStorage та перезавантажує сторінку. Важливо що авторизація зберігається не в централізованому React Context або Redux, а безпосередньо в localStorage, що спрощує архітектуру фронтенду. Компонент ModalMessage.jsx є універсальним модальним вікном для відображення повідомлень про успіх або помилку — він приймає параметри show, onClose, title та message та відображається поверх всього інтерфейсу при show === true.

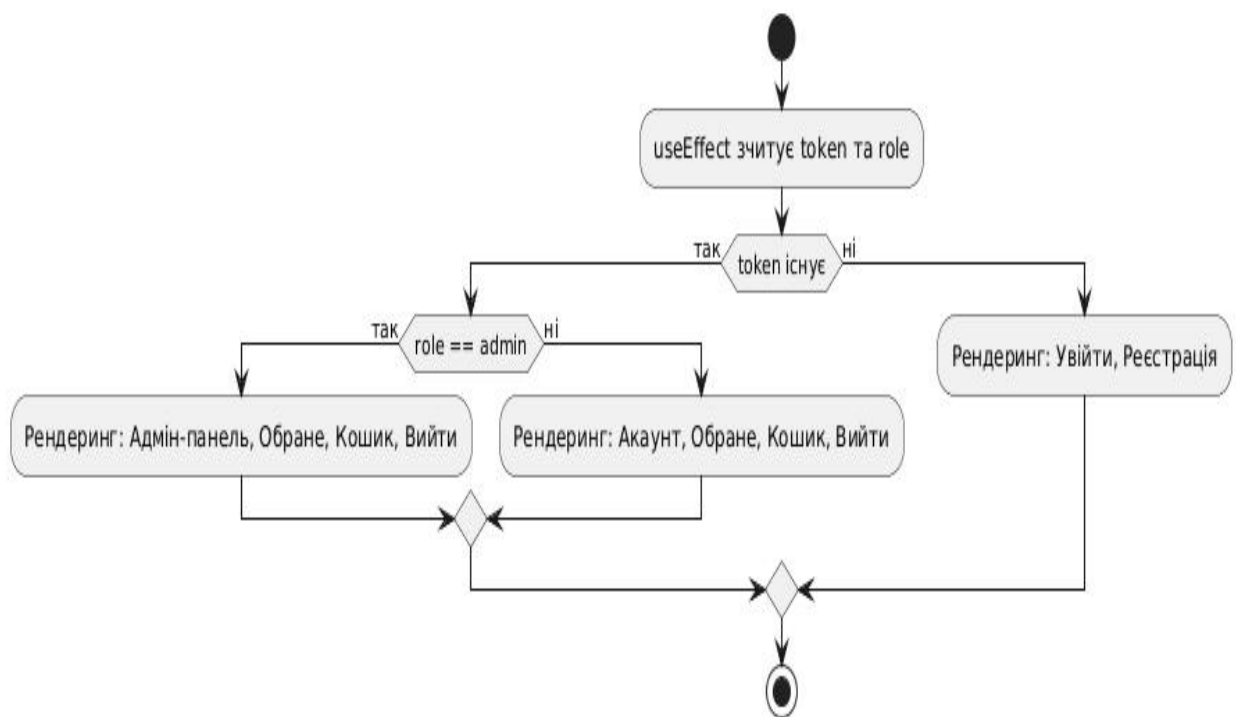


Рис. 2.19 – Схема взаємодії компонентів інтерфейсу та управління станом користувача

Діаграма взаємодії клієнтської частини з серверним API через бібліотеку Axios відображає два сценарії обробки запитів залежно від стану JWT-токена. Процес починається з переходу користувача на захищену сторінку системи,

після чого Frontend зчитує токен з localStorage та формує запит із заголовком Authorization у форматі Bearer token. Сервер отримує запит та виконує валідацію токена через middleware authMiddleware. Діаграма розгалужується на два альтернативних сценарії. У першому сценарії коли токен є валідним сервер виконує запит даних до бази даних MongoDB, отримує відповідь та повертає клієнту HTTP-статус 200 разом із запитаними даними, після чого Frontend відображає відповідний інтерфейс користувача. У другому сценарії коли токен є невалідним або відсутнім сервер повертає HTTP-статус 401 Unauthorized без виконання запиту до бази даних. Frontend отримує статус 401 та встановлює значення параметра show рівним true для компонента ModalMessage, що призводить до відображення модального вікна з повідомленням про необхідність авторизації. Такий підхід забезпечує надійний захист усіх захищених маршрутів системи та коректну обробку помилок автентифікації на рівні клієнтської частини.

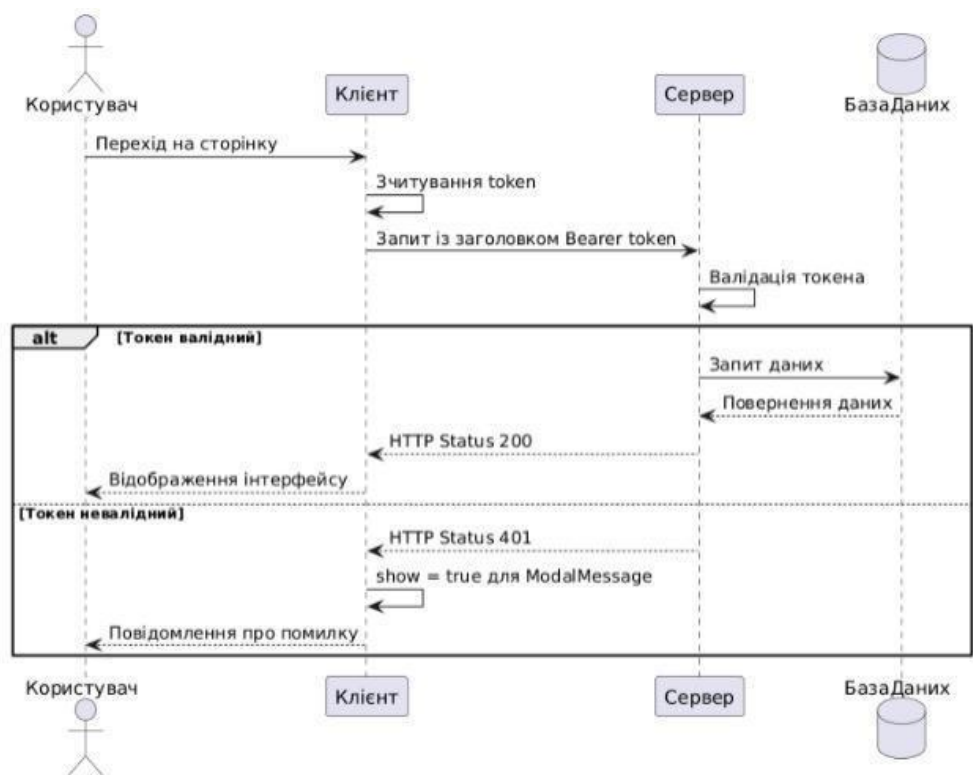


Рис. 2.20 – Діаграма взаємодія через Axios

Висновки до другого розділу

У другому розділі розроблено повну проєктну документацію системи «SkiZone», яка виступає технічним підґрунтям для подальшого кодування. У процесі проєктування було побудовано шість діаграм у нотації PlantUML. Діаграма варіантів використання з трьома акторами (гість, авторизований користувач, адміністратор) та десятима сценаріями дозволила чітко розмежувати права доступу та логіку взаємодії із системою. Три діаграми активності деталізували покрокові алгоритми ключових бізнес-процесів: проходження безпечної аутентифікації, динамічний розрахунок вартості товарів у кошику та багатокритеріальну фільтрацію товарів у каталозі. Побудована діаграма розгортання візуалізувала фізичний розподіл програмних компонентів між клієнтським браузером, вебсервером та хмарним сервером баз даних.

Для організації логіки вебплатформи застосовано архітектурний шаблон MVC (Model-View-Controller). Це дозволило відокремити схеми даних (Models) та логіку відображення інтерфейсу (Views) від безпосередньої бізнес-логіки обробки запитів (Controllers), що спрощує подальше тестування та супровід коду. В межах архітектури задокументовано 19 API-маршрутів системи, розподілених за призначенням: маршрути аутентифікації (/api/auth), керування каталогом товарів (/api/products), обробки кошика (/api/cart), списку бажань (/api/wishlist) та замовлень (/api/orders).

Спроектовано структуру нереляційної бази даних MongoDB, що складається з п'яти взаємопов'язаних колекцій (users, products, carts, orders, wishlists) з детальним описом схем Mongoose. Вибір NoSQL-архітектури обґрунтовано специфікою товарів (лижне спорядження має багато різноманітних атрибутів — ростовка, жорсткість, радіус повороту), що вимагає гнучкої структури документів без жорстких обмежень, притаманних традиційним реляційним СУБД, а також забезпечує високу швидкість виконання операцій читання при високому навантаженні на сайт.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ІНТЕРНЕТ-МАГАЗИНУ ЛИЖНОГО ОБЛАДНАННЯ

3.1 Порядок встановлення та налаштування системи

Система «SkiZone» складається з двох незалежних застосунків — серверної частини skizone-backend та клієнтської частини skizone-frontend, кожна з яких запускається окремо. Для розгортання системи необхідно мати встановлене таке програмне забезпечення: Node.js версії 18 або вище, менеджер пакетів npm, що постачається разом з Node.js, та MongoDB версії 6 або вище — локально або у хмарному сервісі MongoDB Atlas [26].

Перед запуском необхідно клонувати репозиторій проекту з GitHub або розпакувати архів з вихідним кодом. Проект містить дві кореневі директорії: skizone-backend з серверною частиною та skizone-frontend з клієнтською частиною. Кожна директорія містить власний файл package.json з переліком залежностей та скриптами запуску.

Розгортання серверної частини виконується у такій послідовності. Спочатку необхідно відкрити термінал та перейти до директорії серверної частини командою `cd skizone-backend`. Далі встановити всі залежності командою `npm install` — ця команда завантажує всі необхідні пакети, зазначені у файлі package.json, включаючи Express, Mongoose, bcryptjs, jsonwebtoken, Multer та Cors. Після встановлення залежностей необхідно створити файл .env у кореневій директорії skizone-backend та заповнити змінні середовища відповідно до таблиці 3.1.

Таблиця 3.1

Змінні середовища серверної частини (.env)

Змінна	Приклад значення	Опис
PORT	5001	Порт запуску Express-сервера
MONGODB_URI	mongodb://localhost:27017/skizone	Рядок підключення до MongoDB
JWT_SECRET	your_secret_key_here	Секретний ключ для підпису JWT-токенів

Після налаштування файлу `.env` сервер запускається командою `npm start` для продакшн-режиму або `npm run dev` для режиму розробки з автоматичним перезапуском при зміні файлів. За необхідності базу даних можна заповнити тестовими товарами командою `node seed.js` — скрипт автоматично створить набір лишнього спорядження різних категорій.

Після створення файлу конфігурації необхідно переконатися, що всі зазначені параметри відповідають локальному середовищу розробки. Особливу увагу слід приділити правильності адреси підключення до MongoDB та значенню секретного ключа JWT, оскільки від цих параметрів залежить коректна робота механізмів автентифікації користувачів і взаємодії із базою даних. У разі використання хмарного сервісу MongoDB Atlas замість локального сервера необхідно вказати відповідний URI підключення.

Розгортання клієнтської частини виконується аналогічно. У новому вікні терміналу необхідно перейти до директорії `cd skizone-frontend` та виконати `npm install` для встановлення залежностей зокрема Next.js, react та axios. Для режиму розробки застосунків запускається командою `npm run dev`, для продакшн-збірки — послідовно `npm run build` та `npm start`.

Після успішного запуску обох частин системи клієнтська частина стає доступною у веббраузері за адресою `http://localhost:3000`, а серверна частина приймає HTTP-запити за адресою `http://localhost:5001`. Усі API-запити клієнтської частини автоматично спрямовуються до бекенду через префікс `/api`.

Під час першого запуску застосунку рекомендується перевірити відсутність помилок у консолі сервера та браузера. Якщо всі залежності встановлені коректно, серверна частина успішно підключається до бази даних MongoDB, а клієнтська частина компілюється без критичних помилок. За наявності помилок необхідно перевірити правильність встановлення пакетів npm, налаштування змінних середовища та відповідність версій Node.js і MongoDB вимогам проєкту.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

Для опису архітектури розгортання системи використовується діаграма розгортання у нотації UML 2.0 (рис. 3.1). Діаграма відображає фізичне розташування компонентів системи та канали їх взаємодії.

Архітектура розгортання побудована за принципом розділення клієнтської та серверної частин, що дозволяє незалежно запускати, тестувати та масштабувати окремі компоненти системи. Обмін даними між ними здійснюється за допомогою REST API, а всі бізнес-операції, пов'язані з обробкою користувацьких запитів, виконуються виключно на сервері. Такий підхід забезпечує кращу безпеку, централізовану обробку даних і спрощує подальший супровід програмного забезпечення.

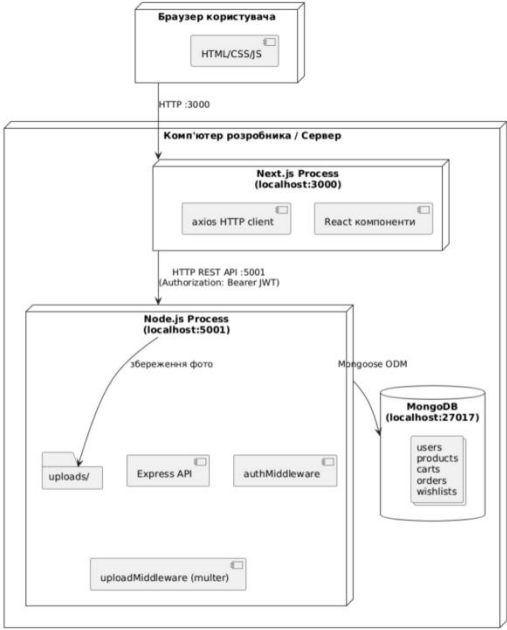


Рис. 3.1 – Діаграма розгортання системи SkiZone

Як показано на рисунку 3.1, основними компонентами архітектури є веббраузер користувача, клієнтський застосунок, серверна частина та база даних MongoDB. Клієнтська частина реалізована із використанням Next.js і відповідає за відображення інтерфейсу користувача, формування HTTP-запитів та обробку отриманих відповідей. Серверна частина реалізована на платформі Node.js із використанням фреймворку Express.js та забезпечує

виконання бізнес-логіки, автентифікацію користувачів, обробку запитів до каталогу товарів, кошика, списку бажань і замовлень.

Для взаємодії із базою даних використовується бібліотека Mongoose, яка забезпечує роботу з документами MongoDB через моделі даних. Усі звернення до бази даних виконуються централізовано серверною частиною, що виключає прямий доступ клієнта до інформації та підвищує рівень захисту даних. Передача інформації між Frontend і Backend здійснюється у форматі JSON через REST API, що дозволяє легко масштабувати систему та інтегрувати її з іншими сервісами в майбутньому.

Запропонована схема розгортання є достатньо простою для локальної розробки та тестування, водночас вона може бути адаптована для розміщення у хмарному середовищі. У такому випадку клієнтська та серверна частини можуть бути розгорнуті на окремих серверах або контейнерах, а база даних — перенесена до хмарного сервісу MongoDB Atlas без необхідності внесення суттєвих змін до програмного коду.

3.2 Опис інтерфейсу користувача

Skizone — це дипломний e-commerce проєкт з клієнтським інтерфейсом на Next.js та серверною частиною на Node.js/Express, який працює з MongoDB для збереження даних. Проєкт надає повний набір функцій онлайн-магазину: реєстрацію, авторизацію, перегляд каталогу товарів, пошук і фільтрацію, кошик з можливістю зміни кількості, оформлення замовлення, список бажань та адміністративну панель для керування товарами. Інтерфейс побудований як SPA-клієнт, що працює через REST API, а бекенд обробляє запити, авторизацію JWT і зберігає зв'язані документи User, Product, Cart, Order та Wishlist. Така архітектура дозволяє розділити рендеринг, бізнес-логіку та збереження даних, забезпечуючи зручність користування і простоту подальшого масштабування.

Навігаційна панель NavBar.jsx відображається на всіх сторінках застосунку і динамічно змінює свій вміст залежно від стану авторизації: для

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

неавторизованих користувачів видно лише посилання "Увійти" та "Реєстрація", тоді як після аутентифікації, на основі значень token та role, що зчитуються з localStorage, інтерфейс адаптується під роль користувача — для ролі user з'являються посилання "Акаунт", "Обране" та "Кошик", а для ролі admin замість "Акаунт" показується "Адмін-панель" з можливістю доступу до CRUD-операцій; при цьому кнопка виходу одразу видаляє токен і роль із сховища та оновлює маршрут, що гарантує коректне відображення навігації в реальному часі й унеможливлює несанкціонований доступ до захищених маршрутів.

Головна сторінка SKIZONE складається з кількох секцій: перша — великий банер-герой з фотографією лижника, назвою SKIZONE та слоганом; друга — блок фільтрації та пошуку товарів, що дозволяє одночасний пошук за назвою та описом, фільтрацію за категорією та сортування за різними критеріями; третя — адаптивна сітка карток товарів з інформацією про ціни й зручними кнопками додавання в кошик; четверта — блок "Про нас" з інформацією про магазин; п'ята — футер з контактною інформацією та формою зворотного зв'язку. Ми гарантуємо швидку доставку та професійну підтримку клієнтів.

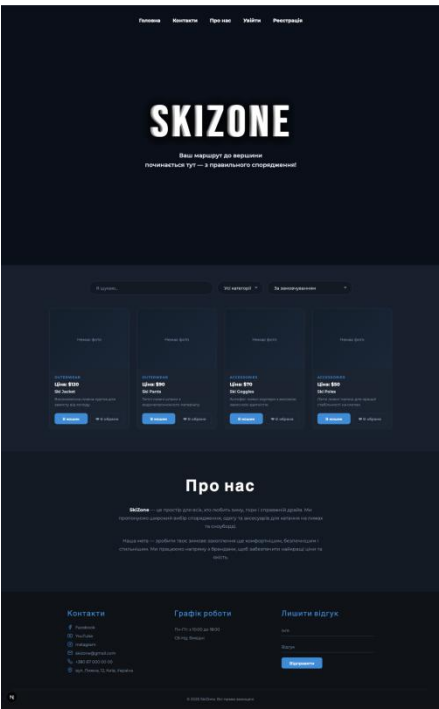


Рис. 3.1 – Головна сторінка магазину SkiZone

Процес авторизації реалізований наступним чином. Після введення логіну та паролю клієнт надсилає POST-запит на `/api/users/login`. Сервер знаходить користувача за полем `login` у колекції `users` та перевіряє пароль через `bcrypt.compare`. У разі невідповідності повертається код 401 та відображається модальне вікно з повідомленням "Невірний логін або пароль". При успішній перевірці генерується JWT-токен з `payload { userId, role }` та терміном дії 1 доба. Клієнт зберігає отриманий токен та роль у `localStorage` та перенаправляє користувача на головну сторінку. Наявність токена у `localStorage` визначає чи є користувач авторизованим при наступних відвідуваннях сайту.

Картка товару відображає фотографію, категорію, ціну та назву. Кнопка «В кошик» надсилає POST `/api/cart/add` із заголовком `Authorization`. Кнопка «В обране» надсилає POST `/api/wishlist/add`. При спробі неавторизованого користувача система відображає модальне вікно `ModalMessage` з повідомленням про необхідність авторизації.

Сторінка авторизації (рис. 3.2) містить форму з полями «Логін» та «Пароль» та кнопку «Увійти». При успіху сервер повертає JWT-токен та роль; клієнт зберігає їх у `localStorage` та перенаправляє на головну сторінку через `window.location.href = "/"`.

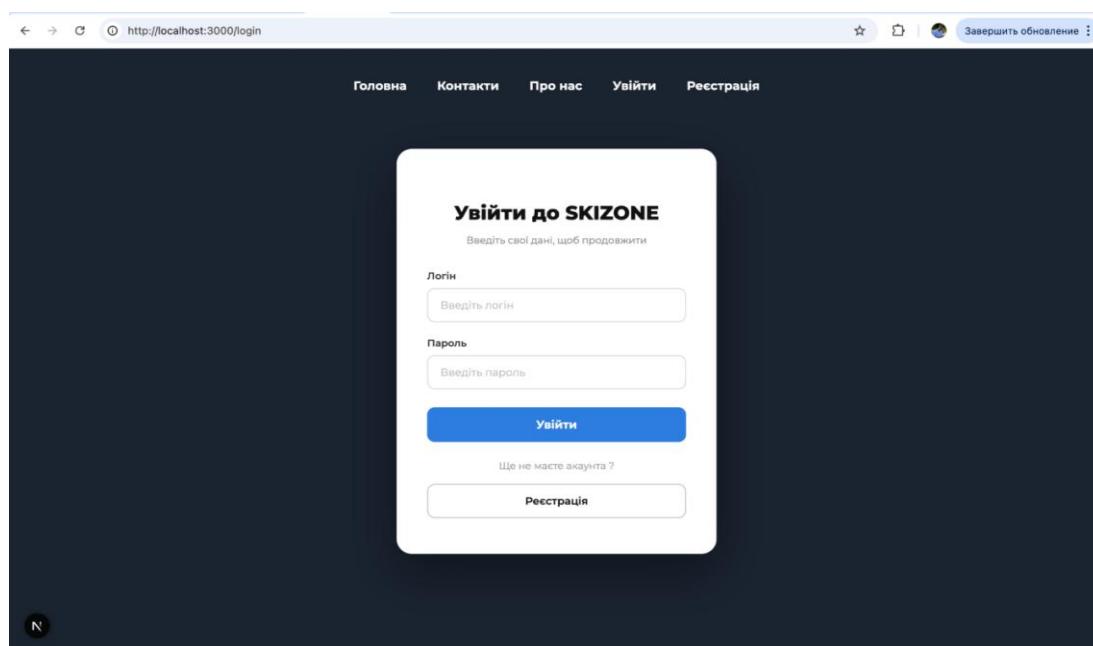


Рис. 3.2 – Сторінки авторизації

Сторінка реєстрації (рис. 3.4) містить форму з полями «Логін» та «Пароль» і кнопку «Зареєструватися». Після успішної реєстрації клієнт надсилає POST-запит на `/api/users/register`, а сервер створює нового користувача в базі даних та повертає повідомлення про успішне створення облікового запису.

Рис. 3.4 – Сторінки реєстрації

Сторінка реєстрації аналогічна до сторінки авторизації. Після успішної реєстрації `ModalMessage` інформує користувача про успіх та пропонує увійти. Функціонал кошика системи `SkiZone` базується на взаємодії клієнтської сторінки (рис. 3.3) із NoSQL-моделлю `Cart` в `MongoDB`. Структура документа кошика містить системний `_id`, посилання на користувача `user` у форматі `ObjectId` та масив об'єктів `items`, де кожен елемент пов'язує ідентифікатор товару `product` із його кількістю `quantity`.

Логіка обробки даних на серверній стороні реалізована через три методи контролера, які суворо захищені проміжним програмним забезпеченням `authMiddleware` й ізольовані на рівні `req.user.userId`. Метод `getCart` знаходить документ кошика поточного клієнта та виконує операцію `.populate('items.product')` для автоматичного отримання актуальних цін, назв та фото з колекції продуктів. Додавання нових позицій забезпечує метод `addToCart`, який приймає параметри `productId` та `quantity`, проводить валідацію на додатне значення кількості, а потім через функцію `findIndex()` перевіряє наявність товару в масиві. При виявленні збігу ідентифікаторів метод просто інкрементує кількість, а при відсутності позиції — додає новий

об'єкт до масиву та викликає функцію збереження `cart.save()`. Для очищення або повного видалення товару з кошика використовується метод `removeFromCart`, який отримує `productId` через URL-параметр запиту `DELETE /api/cart/remove/:id` та відфільтровує масив `items` за допомогою стандартного методу `filter()`.

На фронтенді сума кошика розраховується в режимі реального часу ітерацією по масиву товарів за формулою:

$$\text{Total} = \sum (\text{product.price} \times \text{item.quantity}), \quad (3.1)$$

де `Total` – загальна вартість кошика покупця;

`product.price` – ціна конкретного товару в базі даних;

`item.quantity` – кількість одиниць цього товару, додана користувачем.

При натисканні кнопки «Оформити замовлення» відкриється модальна форма. У разі вибору безготівкового розрахунку клієнтський скрипт активує поля карти та блокує відправку форми до повного проходження регулярної валідації, яка перевіряє наявність 16 цифр номера, відповідність формату дати ММ/YY та наявність 3 або 4 цифр CVV-коду. Після успішної перевірки запит `POST /api/orders/create` передає керування до `orderController`. Сервер повторно обчислює фінальну вартість для захисту від підміни ціни на клієнтській стороні, копіює масив `items` у новий документ колекції `Order` та очищає поточний кошик користувача шляхом присвоєння `cart.items = []`, що повністю забезпечує транзакційність і захист від повторних замовлень.

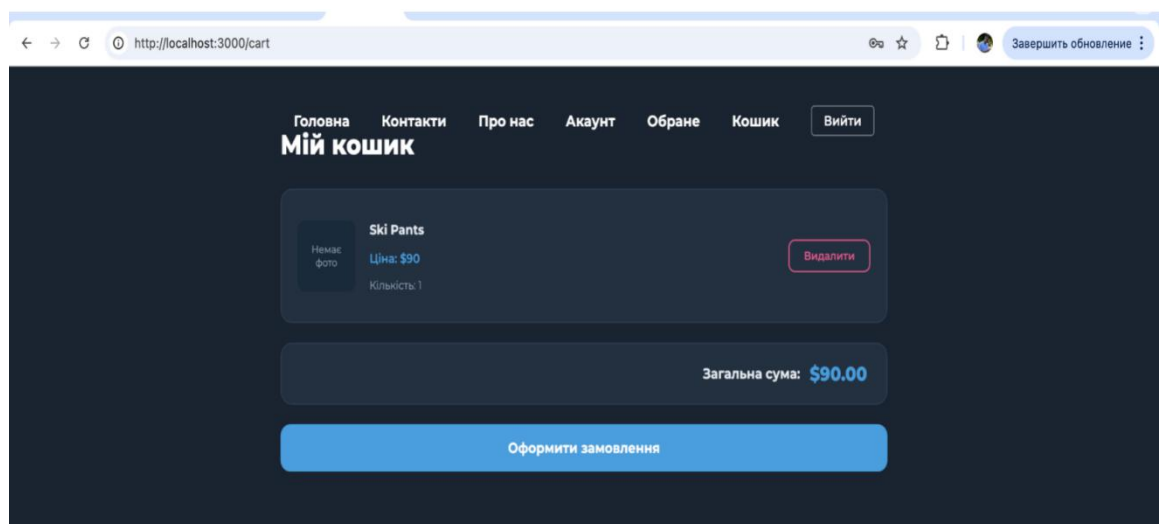


Рис. 3.4 – Сторінка кошика та форма оформлення замовлення

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

Сторінка обраного відображає товари зі списку бажань. Кнопка «В кошик» переміщає товар. Кнопка «Видалити» надсилає DELETE /api/wishlist/remove/:productId. Передбачено також очищення всього списку через DELETE /api/wishlist/clear.

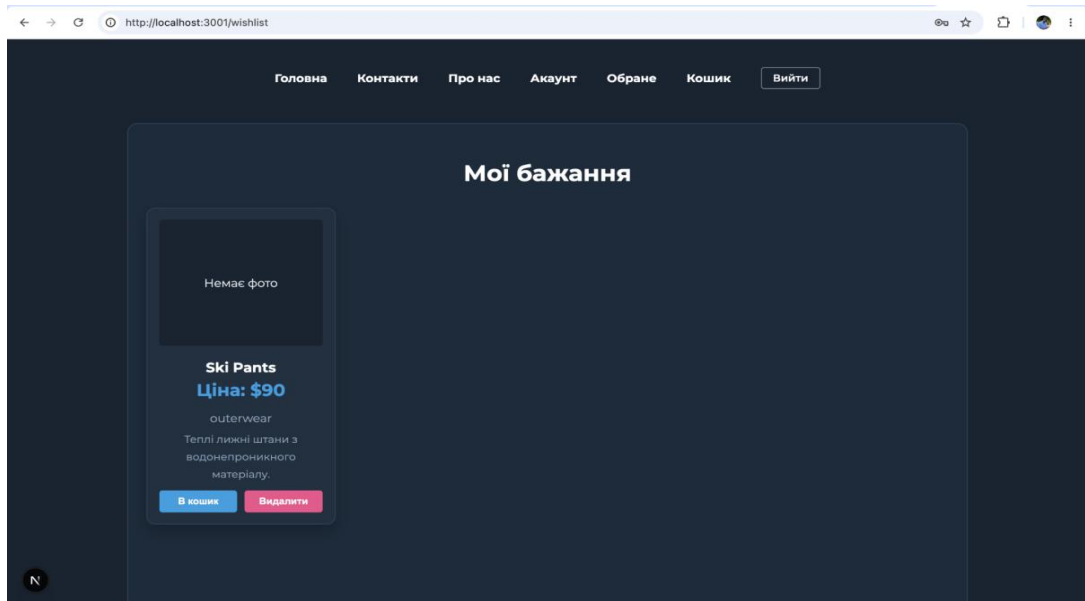


Рис. 3.5 – Сторінка списку бажань

Особистий кабінет містить форму оновлення облікових даних (PATCH /api/users/update) та секцію «Мої замовлення» зі списком замовлень, отриманих через GET /api/orders/my.

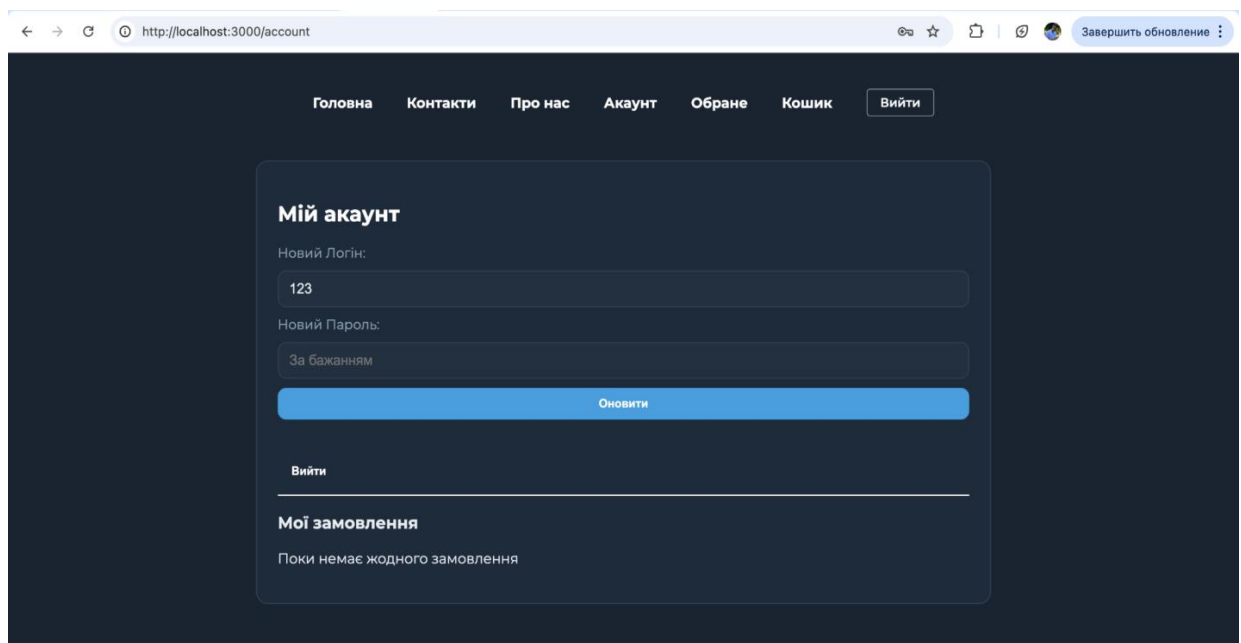


Рис. 3.6 – Сторінка особистого кабінету

Секція "Мої замовлення" відображає повну історію покупок

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

користувача отриману через GET /api/orders/my. Замовлення сортуються за датою у спадному порядку — найновіші відображаються першими. Для кожного замовлення показується дата оформлення, загальна сума та поточний статус. Можливі статуси замовлення: "Processing" — в обробці адміністратором, "Shipped" — товар відправлено, "Delivered" — товар доставлено покупцю.

Адміністративна панель доступна лише при role === "admin" у localStorage. Містить два розділи: управління товарами та замовленнями. Форма додавання товару включає поля назва, опис, ціна, категорія та кнопку вибору файлу (upload.single("image")). Файл зберігається у uploads/ з ім'ям у форматі Date.now() + "-" + random + extension.

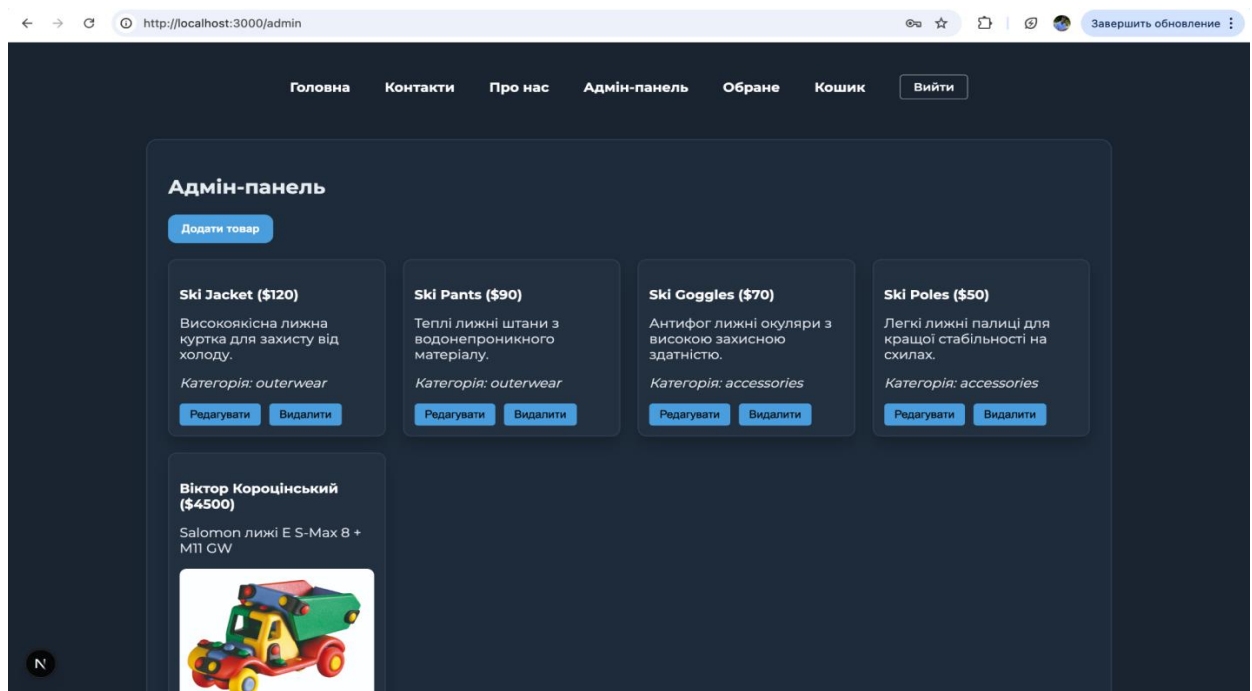


Рис. 3.7 – Адміністративна панель

При завантаженні фотографії товару middleware multer генерує унікальне ім'я файлу у форматі Date.now() + "-" + random + розширення та зберігає його у директорії uploads/. Шлях до файлу записується у поле imagePath документа товару в MongoDB. Статична роздача через app.use('/uploads', express.static('uploads')) дозволяє клієнту отримувати зображення за URL виду <http://localhost:5001/uploads/filename.jpg>. Форма редагування товару дозволяє оновити будь-яке поле включаючи заміну

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

фотографії — при завантаженні нового файлу поле imagePath оновлюється новим шляхом.

Сторінка деталей товару відображається при натисканні на картку у каталозі та містить розширену інформацію: велике зображення товару, назву, категорію, ціну та детальний опис. Кнопки "В кошик" та "В обране" дублюють функціональність каталогу. Нижня секція відображає до чотирьох схожих товарів тієї ж категорії — виключаючи поточний товар — що допомагає покупцю знайти альтернативи або додаткове спорядження.

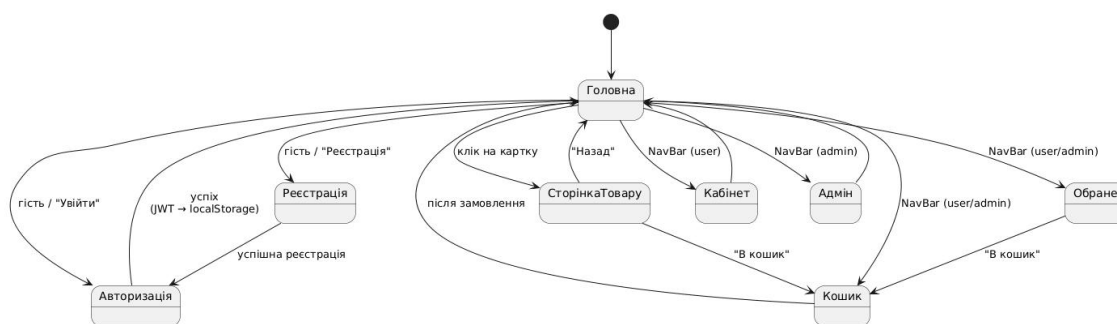


Рис. 3.8 – діаграма переходів відображення навігації між сторінками системи

3.3 Тестування програмного комплексу

Для перевірки коректності роботи системи SkiZone проведено функціональне тестування основних сценаріїв використання. Метод тестування — ручне тестування за тестовими сценаріями у браузері Google Chrome із моніторингом запитів через інструменти розробника Chrome DevTools. Верифікація програмного комплексу охопила двадцять базових сценаріїв взаємодії, розділених на позитивні та негативні тест-кейси.

При введенні унікального логіну та паролю форма реєстрації успішно створює нового користувача, тоді як спроба створення дубліката блокується сервером із поверненням коду 400. Успішна автентифікація завершується генерацією JSON Web Token та його збереженням у localStorage, а спроба входу з невірним паролем генерує помилку 401 та виводить вікно через уніфікований компонент ModalMessage [28]. Пошук, вибір категорій та активація сортування забезпечують правильне динамічне упорядкування

карток товарів, а клік на окрему картку перенаправляє користувача на динамічний маршрут /product/:id.

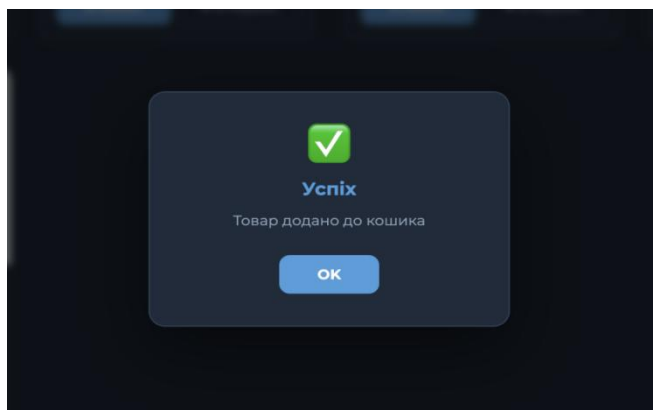


Рис. 3.9 – Компонент ModalMessage у дії

Табл. 3.2

Результати функціонального тестування системи SkiZone

№	Функція	Тестовий сценарій	Очікуваний результат	Статус
1	Реєстрація	Введення унікального логіну та паролю	Код 201, акаунт створено	✓ Пройдено
2	Реєстрація (дублікат)	Введення існуючого логіну	Код 400, "Користувач вже існує"	✓ Пройдено
3	Авторизація	Введення правильних логіну/паролю	JWT у localStorage, redirect на /	✓ Пройдено
4	Авторизація (помилка)	Невірний пароль	Код 401, ModalMessage про помилку	✓ Пройдено
5	Пошук товарів	Введення тексту у поле пошуку	Фільтрація за name та description	✓ Пройдено
6	Фільтр за категорією	Вибір категорії зі списку	Товари обраної категорії	✓ Пройдено
7	Сортування	Вибір "ціна від дешевих"	Товари за зростанням ціни	✓ Пройдено

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

8	Сторінка товару	Клік на картку	Перехід на /product/:id	✓ Пройдено
9	Додавання до кошика	"В кошик" (авторизований)	POST /api/cart/add, товар у кошику	✓ Пройдено
10	Повторне додавання	"В кошик" для того ж товару	quantity збільшується на 1	✓ Пройдено
11	Кошик без авторизації	"В кошик" (гість)	ModalMessage "Увійдіть у систему"	✓ Пройдено
12	Видалення з кошика	Кнопка "Видалити"	Товар видалено з кошика	✓ Пройдено
13	Оформлення замовлення	Заповнення форми + підтвердження	Замовлення в БД, кошик очищено	✓ Пройдено
14	Валідація картки	Невірний номер картки	Форма заблокована, помилка валідації	✓ Пройдено
15	Список бажань	"В обране" (авторизований)	POST /api/wishlist/add	✓ Пройдено
16	Дублювання у вішлісті	Повторне "В обране"	Код 400, товар вже у списку	✓ Пройдено
17	Очищення вішліста	Кнопка очищення	DELETE /api/wishlist/clear	✓ Пройдено
18	Оновлення профілю	Зміна логіну в кабінеті	PATCH /api/users/update, дані оновлено	✓ Пройдено
19	Додавання товару (адмін)	Форма + завантаження фото	Файл в uploads/, товар у БД	✓ Пройдено
20	Захист API	GET /api/cart без токена	HTTP 401 Unauthorized	✓ Пройдено

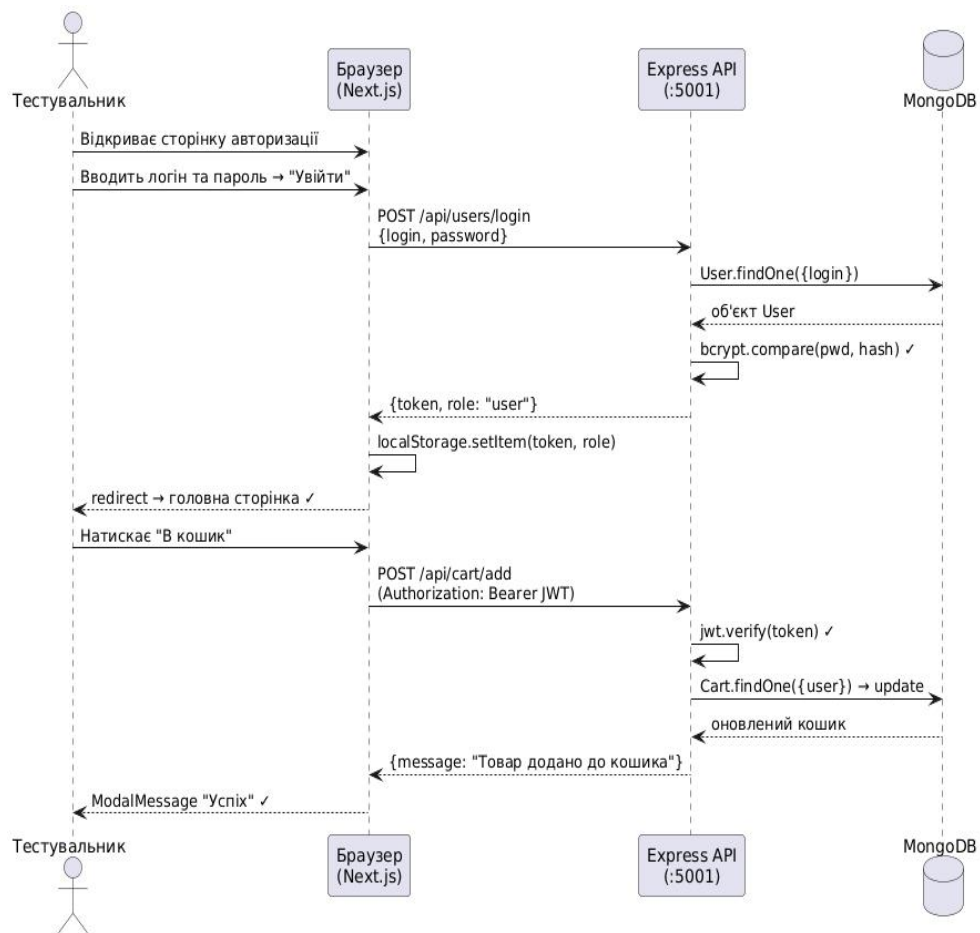


Рис. 3.10 – Діаграма відображення процесу тестування

Висновки до третього розділу

У третьому розділі кваліфікаційної роботи детально описано процес розгортання та практичної реалізації інтернет-магазину лижного обладнання «SkiZone». Розроблену систему успішно запущено в локальному середовищі з чітким розділенням на клієнтську частину під управлінням Next.js та серверну на базі Node.js і Express. У ході роботи було повністю налаштовано конфігурацію змінних середовища у файлі конфігурації, що забезпечило надійне підключення до бази даних MongoDB та безпечне збереження ключів для генерації токенів доступу.

Окрему увагу приділено повній програмній реалізації та візуальному оформленню інтерфейсу користувача. Описано логіку роботи головної сторінки з динамічною фільтрацією та пошуком товарів, модулів реєстрації й

автентифікації користувачів за допомогою захищених JWT-токенів, а також системи кошика, яка автоматично розраховує фінальну вартість замовлення та валідує платіжні реквізити. Окрім цього, розроблено функціонал списку бажань, особистий кабінет із переглядом історії покупок та адміністративну панель для зручного керування асортиментом і завантаження фотографій через спеціальні модулі обробки даних.

Для перевірки стабільності та надійності створеної платформи було проведено комплексне функціональне тестування бізнес-логіки сайту та захищеності його маршрутів. Інженерна перевірка охопила 20 базових сценаріїв використання, серед яких тестувалися як стандартні дії покупця, так і потенційні помилки чи спроби несанкціонованого доступу до бази даних без авторизації, з урахуванням рекомендацій щодо безпеки вхідних даних. Усі розроблені тести було пройдено повністю успішно, а виявлені помилки вводу коректно оброблялися системою з виведенням зрозумілих повідомлень для користувача, що підтверджує технічну готовність інтернет-магазину до реальної експлуатації.

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра спроектовано та розроблено повнофункціональний інтернет-магазин лижного обладнання «SkiZone» на основі сучасного технологічного стеку: Next.js 15 для клієнтської частини, Node.js 18 та Express.js 4 для серверної частини та MongoDB 6 для зберігання даних.

У першому розділі проведено аналіз предметної галузі електронної комерції у сфері лижного спорядження. Встановлено що ринок онлайн-торгівлі в Україні щороку зростає на 25–30%, а попит на лижне спорядження має виражену сезонність з піком у жовтні–березні. Досліджено п'ять аналогів програмного продукту: Extremstyle, Unisport, Keeperstore, Decathlon та Rozetka. На основі порівняльного аналізу визначено конкурентні переваги розробленої системи: спеціалізація на лижному обладнанні, JWT-автентифікація з розмежуванням ролей та список бажань з функцією повного очищення. Обґрунтовано вибір технологічного стеку та сформовано функціональні й нефункціональні вимоги до системи.

У другому розділі розроблено повну проектну документацію системи. Побудовано діаграму варіантів використання з трьома акторами (Гість, Користувач, Адміністратор) та дев'ятьма сценаріями взаємодії. Спроектовано структуру бази даних MongoDB з п'яти колекцій: users, products, carts, orders та wishlists з детальним описом схем Mongoose. Побудовано комплекс UML-діаграм у нотації PlantUML: діаграму варіантів використання, ER-діаграму, діаграми активності для алгоритмів аутентифікації, розрахунку суми кошика та фільтрації товарів, діаграму класів, діаграми послідовності та діаграму розгортання. Описано MVC-архітектуру серверної частини, задокументовано 19 API-маршрутів системи та детально описано логіку роботи кожного контролера.

У третьому розділі детально описано процес розгортання системи, а також надано покрокові інструкції щодо запуску її серверної та клієнтської частин. У межах цього етапу було детально охарактеризовано інтерфейс усіх

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

сторінок системи, зокрема головну сторінку з інтегрованою динамічною фільтрацією каталогу, форми авторизації та реєстрації користувачів, модуль кошика з обов'язковою валідацією введених реквізитів банківської картки, список бажань, особистий кабінет клієнта та спеціалізовану адміністративну панель. Для підтвердження стабільності та надійності вебплатформи проведено функціональне тестування двадцяти сценаріїв роботи, за результатами якого всі тести було пройдено успішно, включаючи ретельну перевірку негативних сценаріїв взаємодії та захисту програмного інтерфейсу API.

У результаті виконання роботи було досягнуто низки важливих практичних результатів. Зокрема, успішно реалізовано каталог товарів із можливістю повнотекстового пошуку за назвою й описом, гнучкою фільтрацією за категоріями та динамічним сортуванням асортименту. Також було розроблено надійну систему JWT-аутентифікації з передачею корисного навантаження типу `payload {userId, role}` та встановленим терміном дії токена в одну добу. Програмно реалізовано модуль кошика з алгоритмом автоматичного збільшення кількості продукції при її повторному додаванні, а також створено підсистему оформлення замовлення з підтримкою двох альтернативних способів оплати та вбудованою валідацією даних. Разом з тим розроблено модуль списку бажань із забезпеченням унікального індексу для кожної позиції та функцією повного очищення переліку, спроектовано особистий кабінет із можливістю редагування профілю й перегляду хронології замовлень, а також впроваджено адміністративну панель із підтримкою повного циклу CRUD-операцій та механізмом завантаження фотографій через проміжне програмне забезпечення Multer, що підкріплено надійним захистом маршрутів за допомогою `authMiddleware` та `adminMiddleware`.

Розроблений інтернет-магазин «SkiZone» є повноцінним програмним продуктом, повністю готовим до безпосереднього розгортання та комерційної експлуатації. Подальший розвиток і масштабування створеної

					ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

системи може включати інтеграцію з міжнародним платіжним шлюзом Stripe, проектування та розробку кросплатформеного мобільного додатка для користувачів, а також впровадження інтелектуальної системи рекомендацій товарів на основі аналізу історії попередніх переглядів.

					ПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Laudon K. C., Traver C. G. E-Commerce 2022: Business, Technology, Society. 17th ed. New York : Pearson, 2021. 912 p. URL: <https://www.pearson.com/en-us/subject-catalog/p/e-commerce-2022-business-technology-society/P200000003276> (дата звернення: 01.03.2026).
2. Turban E., Outland J., King D. Electronic Commerce 2018: A Managerial and Social Networks Perspective. Springer Texts in Business and Economics. 2018. Vol. 9, No 1. P. 3–54. DOI: 10.1007/978-3-319-58715-8.
3. Спортивний інтернет-магазин Extremstyle.ua : веб-сайт. URL: <https://extremstyle.ua/> (дата звернення: 01.03.2026).
4. Інтернет-магазин спортивного обладнання Unisport.ua : веб-сайт. URL: <https://unisport.ua/> (дата звернення: 01.03.2026).
5. Інтернет-магазин спортивного спорядження Keeperstore.ua : веб-сайт. URL: <https://keeperstore.ua/> (дата звернення: 01.03.2026).
6. Decathlon Ukraine : офіційний веб-сайт компанії. URL: <https://www.decathlon.ua/> (дата звернення: 01.03.2026).
7. Інтернет-магазин Rozetka.ua : веб-сайт. URL: <https://rozetka.com.ua/> (дата звернення: 01.03.2026).
8. Next.js Documentation : official technical documentation : веб-сайт. URL: <https://nextjs.org/docs> (дата звернення: 10.03.2026).
9. Node.js Documentation : official API reference : веб-сайт. URL: <https://nodejs.org/en/docs> (дата звернення: 10.03.2026).
10. Express.js Documentation : fast, unopinionated, minimalist web framework documentation : веб-сайт. URL: <https://expressjs.com/> (дата звернення: 10.03.2026).
11. MongoDB Manual : official database documentation : веб-сайт. URL: <https://www.mongodb.com/docs/manual/> (дата звернення: 10.03.2026).
12. Mongoose ODM Documentation : elegant mongodb object modeling for node.js : веб-сайт. URL: <https://mongoosejs.com/docs/> (дата звернення: 12.03.2026).

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		69

13. Jones M., Bradley J., Sakimura N. JSON Web Token (JWT). RFC 7519. Internet Engineering Task Force, 2015. P. 1–32. URL: <https://www.rfc-editor.org/rfc/rfc7519> (дата звернення: 15.03.2026).
14. Axios : Promise based HTTP client for the browser and node.js : веб-сайт. URL: <https://axios-http.com/docs/intro> (дата звернення: 15.03.2026).
15. Multer : npm library module for handling multipart/form-data : веб-сайт. URL: <https://www.npmjs.com/package/multer> (дата звернення: 15.03.2026).
16. React Documentation : official reference for building user interfaces : веб-сайт. URL: <https://react.dev/> (дата звернення: 16.03.2026).
17. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : PhD dissertation. Irvine : University of California, 2000. 182 p. URL: <https://ics.uci.edu/~fielding/pubs/dissertation/top.htm> (дата звернення: 18.03.2026).
18. Chacon S., Straub B. Pro Git : digital book. URL: <https://git-scm.com/book/en/v2> (дата звернення: 18.03.2026).
19. Banks A., Porcello E. Learning React. 2nd ed. Sebastopol : O'Reilly Media, 2020. 310 p. URL: <https://www.oreilly.com/library/view/learning-react-2nd/9781492051718/> (дата звернення: 16.03.2026).
20. Hahn E. Express in Action. Shelter Island : Manning Publications, 2016. 256 p. URL: <https://www.manning.com/books/express-in-action> (дата звернення: 16.03.2026).
21. Wilson M. Node.js 8 the Right Way. Raleigh : Pragmatic Bookshelf, 2018. 334 p. URL: <https://pragprog.com/titles/mwknode/node-js-8-the-right-way/> (дата звернення: 16.03.2026).
22. Bradshaw S., Brazil E., Chodorow K. MongoDB: The Definitive Guide. 3rd ed. Sebastopol : O'Reilly Media, 2019. 514 p. URL: <https://www.oreilly.com/library/view/mongodb-the-definitive/9781492054122/> (дата звернення: 16.03.2026).

23. Pasquali S. Mastering Node.js. Birmingham : Packt Publishing, 2013. 346 p.
URL: <https://www.packtpub.com/product/mastering-node-js/9781782166320> (дата звернення: 16.03.2026).
24. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 533 p. URL: <https://martinfowler.com/books/ea.html> (дата звернення: 18.03.2026).
25. JSON Web Token Introduction : open standard specification : веб-сайт.
URL: <https://jwt.io/introduction> (дата звернення: 20.03.2026).
26. OWASP. Authentication Cheat Sheet : security standards guidelines : веб-сайт.
URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (дата звернення: 22.03.2026).
27. OWASP. SQL Injection Prevention Cheat Sheet : developer security guide : веб-сайт.
URL: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html (дата звернення: 22.03.2026).
28. Almeida F., Monteiro J. A. The Role of Security in Software Development Lifecycles. International Journal of Information Security and Privacy. 2017. Vol. 11, No 4. P. 28–40. DOI: 10.4018/IJISP.2017100103.
29. Shklar L., Rosen R. Web Application Architecture: Principles, Protocols and Practices. 2nd ed. Chichester : John Wiley & Sons, 2009. Vol. 2, No 1. P. 45–112.
URL: <https://www.wiley.com/en-us/Web+Application+Architecture%3A+Principles%2C+Protocols+and+Practices%2C+2nd+Edition-p-9780470518601> (дата звернення: 18.03.2026).
30. Sommerville I. Software Engineering. 10th ed. Boston : Pearson Education, 2016. 816 p. URL: <https://www.pearson.com/en-us/subject-catalog/p/software-engineering/P200000003251> (дата звернення: 18.03.2026).

Додатки

					ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import Layout from '../components/Layout';
import '../styles/globals.css';
import '../styles/navbar.css';
import '../styles/AdminPage.css';
import '../styles/AuthForms.css';
import '../styles/CartPage.css';
import '../styles/HomePage.css';
import '../styles/OrderPage.css';
import '../styles/AccountPage.css';
import '../styles/ModalMessage.css';
import '../styles/WishlistPage.css';
import '../styles/ProductPage.css';
function MyApp({ Component, pageProps }) {
  return (
    <Layout>
    <Component {...pageProps} />
    </Layout>
  );
}

export default MyApp;

```