

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 одягу»

Виконав студент групи ІПЗп-22-1
МІРЕЦЬКИЙ Владислав Ярославович

Керівник роботи:
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

Житомир – 2026

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: **МІРЕЦЬКИЙ Владислав Ярославович**

Керівник роботи: **ЧИЖМОТРЕ Олена Геннадіївна**

Тема роботи: **«Інтернет-магазин одягу»,**

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

Вихідні дані для роботи: **процес розробки інтернет-магазину одягу, з використанням технологій React, Node.js та MongoDB, для здійснення подальшого продажу на платформі.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О. Г.	16.07.2026 р.	16.07.2026 р.
2	Чижмотря О. Г.	16.07.2026 р.	16.07.2026 р.
3	Чижмотря О. Г.	16.07.2026 р.	16.07.2026 р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебдодатку інтернет-магазину одягу та пояснювальної записки. Робота складається зі вступу, 3-х розділів, висновків, списку використаних джерел із 31 найменування (3 стор.), 2 додатків (13 стор.), 22 таблиць та 41 ілюстрації по тексту. Повний обсяг роботи становить 85 сторінок, з них 66 сторінок основного тексту.

Метою роботи є створення сучасного інтернет-магазину одягу завдяки використанню сучасних вебтехнологій а також забезпечити зручне використання ресурсу користувачем.

Архітектура вебдодатку спроектована за клієнт-серверною моделлю, завдяки чому забезпечується швидка обробка запитів та надійне збереження даних про замовлення та користувачів. Реалізовано систему пошуку та фільтрації товарів, керування товарами, кошик, оформлення замовлень та особистий кабінет користувача.

Додаток має адаптивний інтерфейс, який дозволяє зручно здійснювати пошук та користуватися ним з будь-якого пристрою.

Ключові слова: ВЕБЗАСТОСУНОК, ІНТЕРНЕТ-МАГАЗИН, MONGODB, NODEJS, REACT.

					ІПЗ.КР.Б – 121 – 26 – ІЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Інтернет-магазин одягу Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		Мірецький Вл. Я.					3	85
Керівник		Чижмотря О. Г.						
Рецензент		Вакалюк Т.А.				Житомирська політехніка група ІПЗп-22-1		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

Bachelor's qualifying thesis consists of a web application for an online clothing store and an explanatory note. The thesis consists of an introduction, three chapters, conclusions, a list of references containing 31 sources (3 pages), 2 appendices (13 pages), 22 tables, and 41 figures throughout the text. The total length of the thesis is 85 pages, including 66 pages of the main text.

The aim of the work is to create a modern online clothing store through the use of modern web technologies , as well as ensuring a convenient user experience.

The architecture of the web application is designed according to the client-server model, which ensures fast request processing and reliable storage of data about orders and users. A product search and filtering system, product management, shopping cart, order placement and personal user account have been implemented.

The application has a responsive interface that allows convenient search and use from any device.

Keywords: MONGODB, NODEJS, ONLINE STORE, REACT, WEB APPLICATION.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ДЛЯ РОЗРОБКИ ІНТЕРНЕТ- МАГАЗИНУ ОДЯГУ	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір архітектури інтернет-магазину одягу.....	16
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	19
Висновки до першого розділу	23
РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ОДЯГУ CAVERTY MALL.....	24
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	24
2.2. Розробка бази даних системи.....	33
2.3. Проєктування та реалізація алгоритмів роботи системи	37
2.4. Реалізація вебзастосунку інтернет-магазину одягу	40
Висновки до другого розділу	45
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ-МАГАЗИНОМ ОДЯГУ CAVERTY MALL	46
3.1. Порядок встановлення та налаштування параметрів системи	46
3.2. Структура інтерфейсу інтернет-магазину одягу Caverty Mall	49
3.3. Тестування роботи програмного комплексу	59
Висновки до третього розділу.....	64
ВИСНОВКИ.....	65
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	67
ДОДАТКИ.....	70

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - інтерфейс, завдяки якому програма може взаємодіяти між різними компонентами [6, 13].

CSS - мова, для задання стилів, кольору, шрифтів та розміру тексту.

DB - база даних.

HTML - мова розмітки вебсторінки.

JSON - формат передання даних між сервером та клієнтом.

JWT - токен, який підтверджує автентифікацію користувача [22].

MongoDB - база даних, у якій зберігається інформація [17].

Node.js - серверне середовище JavaScript на стороні сервера [15].

React - JavaScript-бібліотека для побудови інтерфейсу користувачів [14].

UI - інтерфейс користувача.

URL - адреса в інтернеті.

UX - досвід користувача при взаємодії з програмним продуктом.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижевотря О.Г.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми полягає у тому, що технологій з кожним днем стає все більше у нашому житті, що в свою чергу значно економить наш час, робить процеси зручнішими та ефективнішими. Одним з таких є сфера онлайн-торгівлі, яка потребує сучасної платформи, на якій можна здійснювати продаж різноманітного одягу, що дозволить оптимізувати процеси та охоплювати більшу кількість аудиторії, ніж офлайн.

На даний момент більшість продавців одягу досі не мають власних онлайн-платформ для реалізації своїх товарів, в результаті чого користуються застарілими рішеннями, це призводить до меншої кількості клієнтів. Саме тому розробка сучасного інтернет-магазину з широким асортиментом товарів, системою фільтрації товарів, власним кабінетом покупця є актуальною задачею.

Метою роботи є розробка інтернет-магазину одягу з використанням сучасних технологій, які дозволять автоматизувати процеси продажу, комунікацію продавця з клієнтом та реалізувати зручний інтерфейс для користувачів, який дозволить ефективно працювати з вебдодатком.

Завдання роботи: провести аналіз конкурентів у сфері онлайн торгівлі одягом; визначити основний функціонал системи та вимоги до неї; обрати та обґрунтувати технології для розробки проєкту; спроектувати архітектуру вебдодатку та структуру бази даних; реалізувати серверну частину з REST API для управління товарами, замовленнями та користувачами; реалізувати систему реєстрації та авторизації користувача; розробити користувацький інтерфейс; протестувати робочу систему та усунути помилки.

Об'єктом дослідження є процес взаємодії між продавцем одягу та покупцем у цифровому середовищі.

Предметом дослідження є використання сучасних вебтехнологій для реалізації всього функціоналу системи.

Методи дослідження: аналіз предметної області та існуючих конкурентів; об'єктно-орієнтоване проєктування; компонентний підхід до розробки

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмотря О.Г.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

інтерфейсу.

Практичне значення отриманих результатів. Розроблений інтернет-магазин може бути використаний для продажу одягу в режимі онлайн. Система дозволяє продавцеві: керувати товарами (їх кількістю та характеристикою); додавати нові позиції; відстежувати замовлення; взаємодіяти з покупцями. Покупець у свою чергу отримує зручний та зрозумілий інтерфейс для пошуку, вибору та замовлення необхідного товару з можливістю відстеження його статусу в особистому кабінеті.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмотря О. Г.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ДЛЯ РОЗРОБКИ ІНТЕРНЕТ-МАГАЗИНУ ОДЯГУ

1.1. Постановка задачі

Інтернет-магазин одягу повинен забезпечити зручне використання, як для покупця так і для адміністратора магазину. Платформа повинна включати в себе: каталог товарів з фільтрацією та пошуком; систему реєстрації та авторизації користувачів [22]; кошик; оформлення замовлення; особистий кабінет покупця з можливістю відслідковування статусу замовлення; адміністративну панель.

В результаті повинен бути створений вебсайт на якому адміністратор може керувати товарами(редагувати та видаляти їх), переглядати та обробляти замовлення, керувати користувачами, керувати відгуками. Покупець в свою чергу зможе здійснювати пошук, фільтрацію, оформлення замовлення та залишити відгук.

Одним з основних етапів розробки платформи є розробка дизайну вебсайту. Необхідно розробити загальний дизайн платформи, який буде сучасним, зручним та інтуїтивно зрозумілим для користувачів. Дизайн повинен відповідати тематиці магазину одягу та залишати після себе приємне візуальне враження. Також включає в себе розробку компонентів інтерфейсу та розміщення сторінок.

Далі необхідно побудувати клієнтську частину додатку, яка включає в себе розробку користувацького інтерфейсу за допомогою бібліотеки React [8], завдяки якому покупці зможуть переглядати каталог товарів, користуватися фільтрами за категорією, розміром та ціною, додавати товар у кошик та оформлювати замовлення. Інтерфейс повинен бути адаптивним, зрозумілим та зручним у використанні на будь-якому пристрої.

Побудова серверної частини додатку включає в себе розробку логіки на основі Node.js та Express [1, 16, 32], обробку запитів від клієнтської частини, роботу з базою даних та реалізацію REST API [6]. Сервер повинен обробляти

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

запити користувачів, керування товарами, замовленнями та користувачами системи.

Необхідно реалізувати реєстрацію та авторизацію користувачів у системі. Включає в себе розробку реєстрації нових користувачів та авторизацію уже існуючих користувачів за допомогою JWT токенів [22]. У системі передбачені різні ролі користувач-покупець та адміністратор-продавець, з правами доступу до усього функціоналу інтернет-магазину.

Ключовим етапом є розробка каталогу товарів та пов'язаному з ними функціоналу. Включає в себе розробку каталогу з можливістю фільтрації за категоріями, розміром, кольором та ціною. Також реалізовано сторінку окремого товару з зображенням, вибором розміру, можливістю додавання товару до кошика та залишення відгуку з рейтингом.

Реалізація кошика та оформлення замовлень включає в себе розробку функціоналу кошика з можливістю додавання товарів, зміни кількості та видалення. Також реалізовано сторінку оформлення замовлень з вибором способу доставки через Нову Пошту або Укрпошту [23], з введенням даних отримувача та вибором способу оплати.

Розробка особистого кабінету користувача та панелі адміністратора включає в себе розробку особистого кабінету користувача з можливістю перегляду та скасування своїх замовлень, також редагування власного профілю. Адміністративна панель забезпечує керування товарами, замовленнями та користувачами системи зі зміною їх ролей.

Після завершення розробки буде проведено тестування працездатності системи, інтерфейсу, серверної логіки, правильність роботи всіх функцій та виправлено усі помилки.

Система відгуків та рейтингу товарів включає в себе розробку функціоналу залишення відгуків покупцем на сторінці товару з можливістю виставлення рейтингу у вигляді зірок від 1 до 5. Також реалізовано можливість відповіді на відгуки інших користувачів, редагування та видалення власних відгуків. Середній рейтинг товару перераховується після кожного нового відгуку.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Розробка сторінки "Список бажань" включає в себе розробку функціоналу за допомогою якого покупці можуть залишити побажання щодо тих товарів, які вони хотіли б бачити в асортименті магазину. Користувач може вказати назву товару, категорію, бажаний ціновий діапазон, опис та додати фото для прикладу. Адміністратор в свою чергу бачить усі побажання в панелі адміністратора. Це допоможе краще розуміти потреби покупців та формувати асортимент відповідно до їх запитів.

1.2. Аналіз аналогів програмного продукту

Перед початком розробки інтернет-магазину одягу було проведено аналіз існуючих аналогів. Для порівняння було обрано три інтернет-магазини одягу: Remix, Goldi та Garda. Це допомогло визначити основні переваги та недоліки вже існуючих конкурентів, також визначити вимоги до розроблюваної системи.

Remix - це українська мережа магазинів стильного одягу, яка працює вже понад 15 років (рис. 1.1).

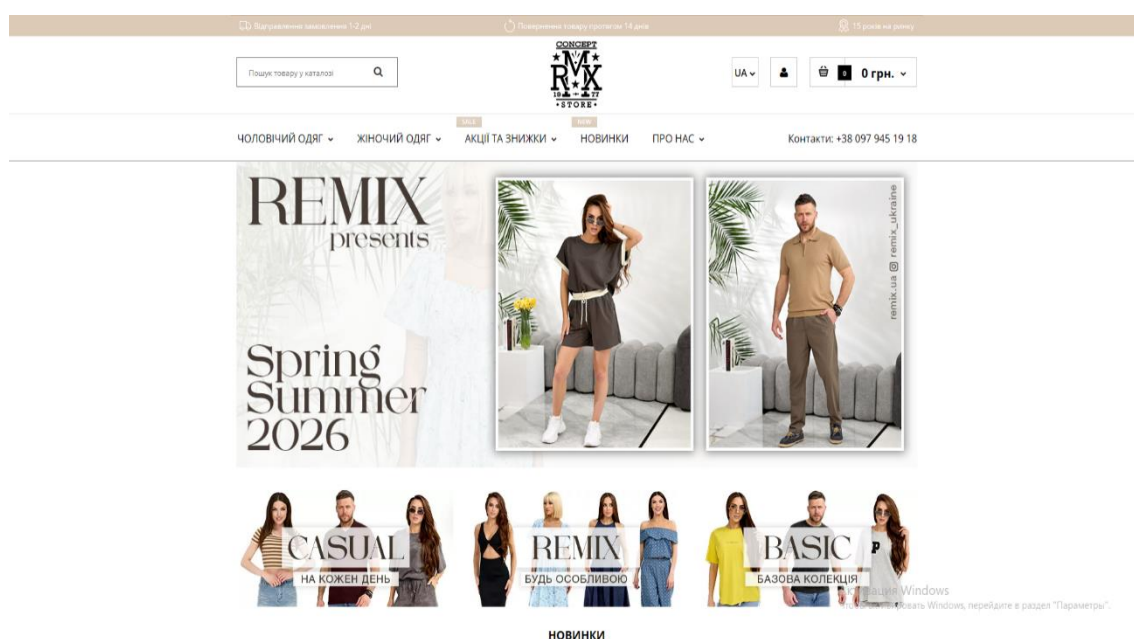


Рис. 1.1 - Головна сторінка інтернет-магазину Remix

Магазин спеціалізується на продажі чоловічого та жіночого одягу власного виробництва, позиціонує себе як доступний бренд для повсякденного використання.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижевотря О.Г.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

На платформі є широкий асортимент чоловічого та жіночого одягу різних категорій: верхній одяг, футболки, сорочки, джинси, шорти та аксесуари.

Сайт має зручний інтерфейс з розширеним каталогом товарів розділеним на чоловічий та жіночий одяг. Присутня система фільтрів за категоріями, пошук товарів, кошик та оформлення замовлень (рис. 1.2).

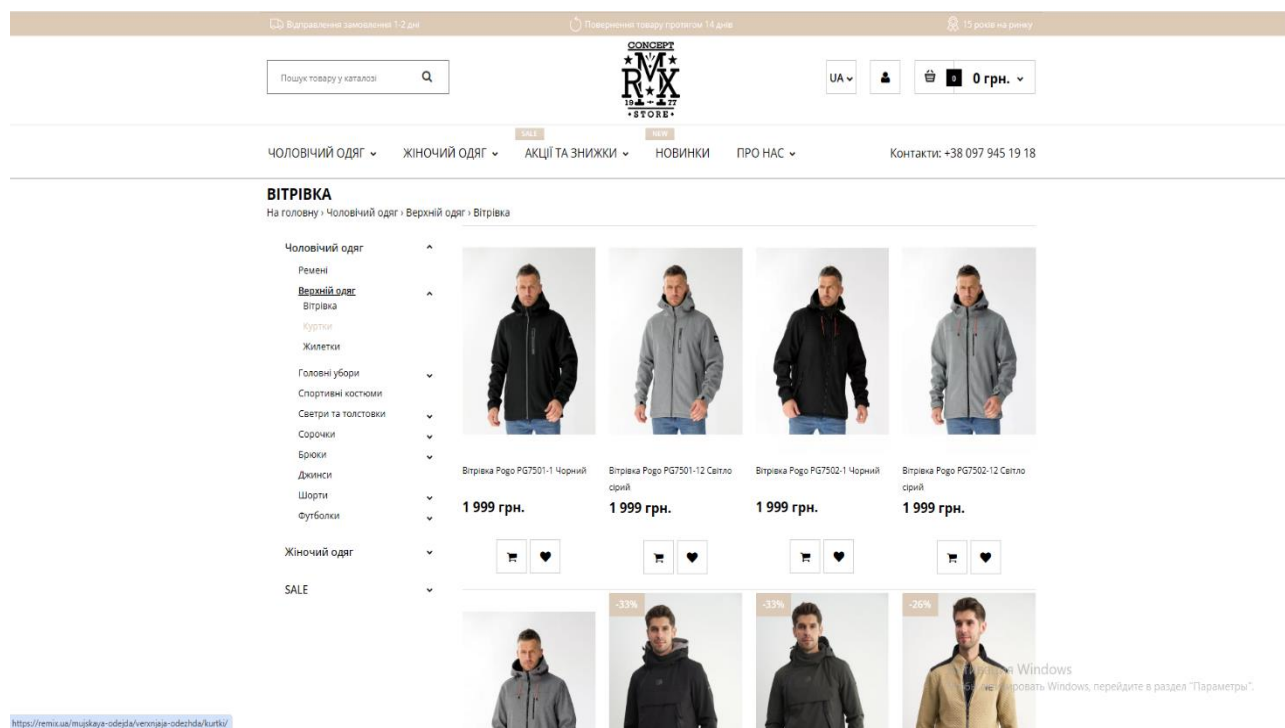


Рис. 1.2 - Сторінка з каталогом товарів інтернет-магазину Remix

Недоліком платформи є обмежена система фільтрації товарів, вона здійснюється лише за загальними категоріями без можливості відфільтрувати за розмірами, кольорами та ціновим діапазоном, також відсутня система відгуків на товари.

Goldi - український бренд одягу заснований у 1990 році. Бренд позиціонує себе як мультикультурне об'єднання, що поєднує світові модні тенденції з українським street-style та робить моду доступною для широкої аудиторії. Інтернет-магазин пропонує жіночий та чоловічий одяг власного виробництва розділений за колекціями та категоріями.

Інтернет-магазин представляє собою сучасний та зручний інтерфейс (рис. 1.3).

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

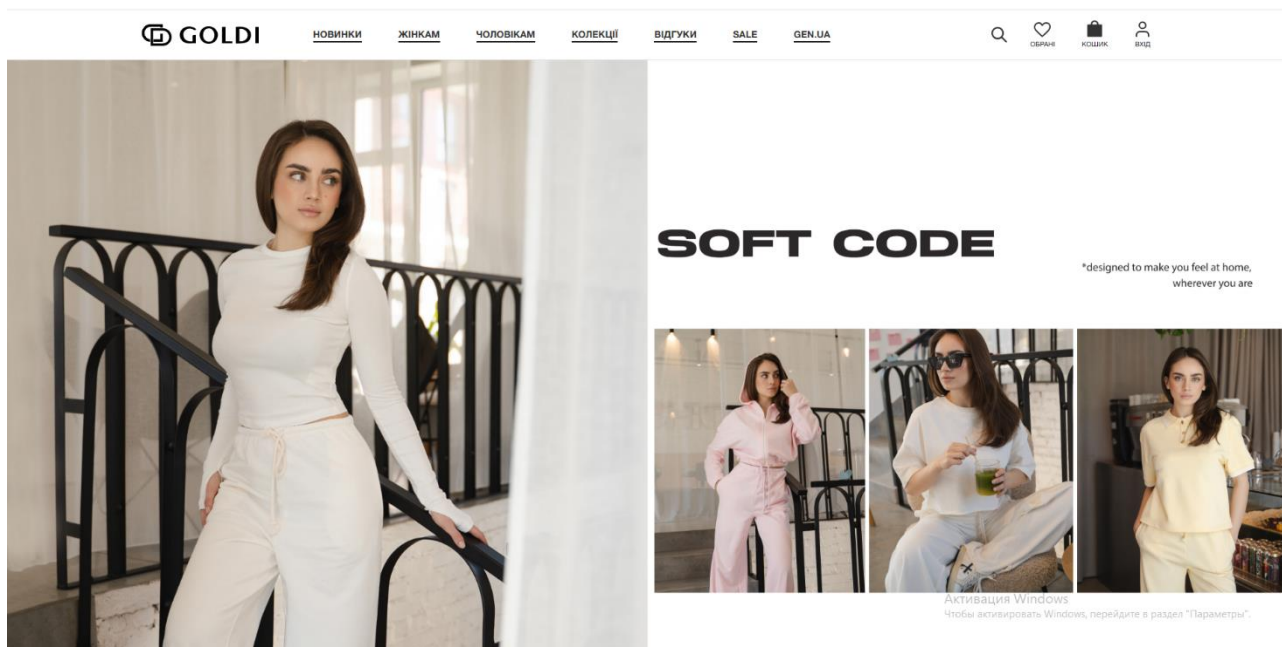


Рис. 1.3 - Головна сторінка інтернет-магазину Goldi

Платформа має широкий функціонал, зокрема окремі розділи для жіночого та чоловічого одягу, сторінка з новинками, новими колекціями та знижками. Також є вкладка обрані товари, що дозволяє покупцеві зберігати вподобані товари для подальшого перегляду. Присутній особистий кабінет користувача зі зручним функціоналом, де є можливість редагувати власний профіль та відслідковувати статус власних замовлень. Каталог товарів має достатньо фільтрів для зручного пошуку необхідного товару (рис. 1.4).

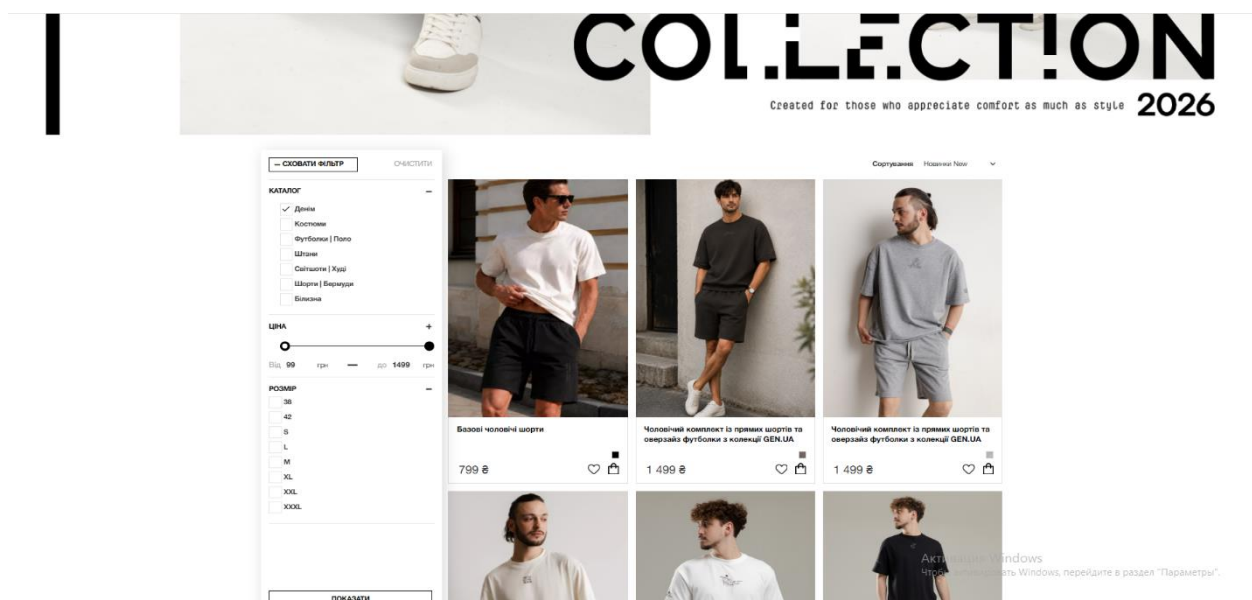


Рис. 1.4 - Сторінка з каталогом товарів інтернет-магазину Goldi

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Недоліками платформи є те, що при застосуванні фільтрів користувач перенаправляється на окрему сторінку, що є доволі незручним. Також система відгуків реалізована окремою вкладкою на сайті, а не на сторінці кожного товару окремо, що доволі ускладнює пошук відгуків до потрібного товару. Відсутня можливість відстеження статусу замовлення.

Garda - український патріотичний бренд одягу. Магазин спеціалізований на одязі з патріотичною символікою та позиціонує себе як "одяг для сильних духом" (рис. 1.5).



Рис. 1.5 - Головна сторінка інтернет-магазину Garda

Інтернет-магазин має стандартний функціонал: каталог товарів розділений на чоловічий та жіночий, кошик та загальний пошук по сайту. Фільтрація товарів дуже стисла містить лише загальні назви без фільтрації за ціновим діапазоном та розміром (Рис. 1.6).

Картка товару достатньо розширена є розмірна таблиця, характеристики, інформація про тканину, друк та доставку. Платформа призначена під конкретну цільову аудиторію та підтримує фірмовий стиль бренду.

Основними недоліками є стисла фільтрація товарів, відсутність особистого кабінету покупця та можливості відстеження статусу замовлення, також відсутня система відгуків.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижевотря О.Г.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

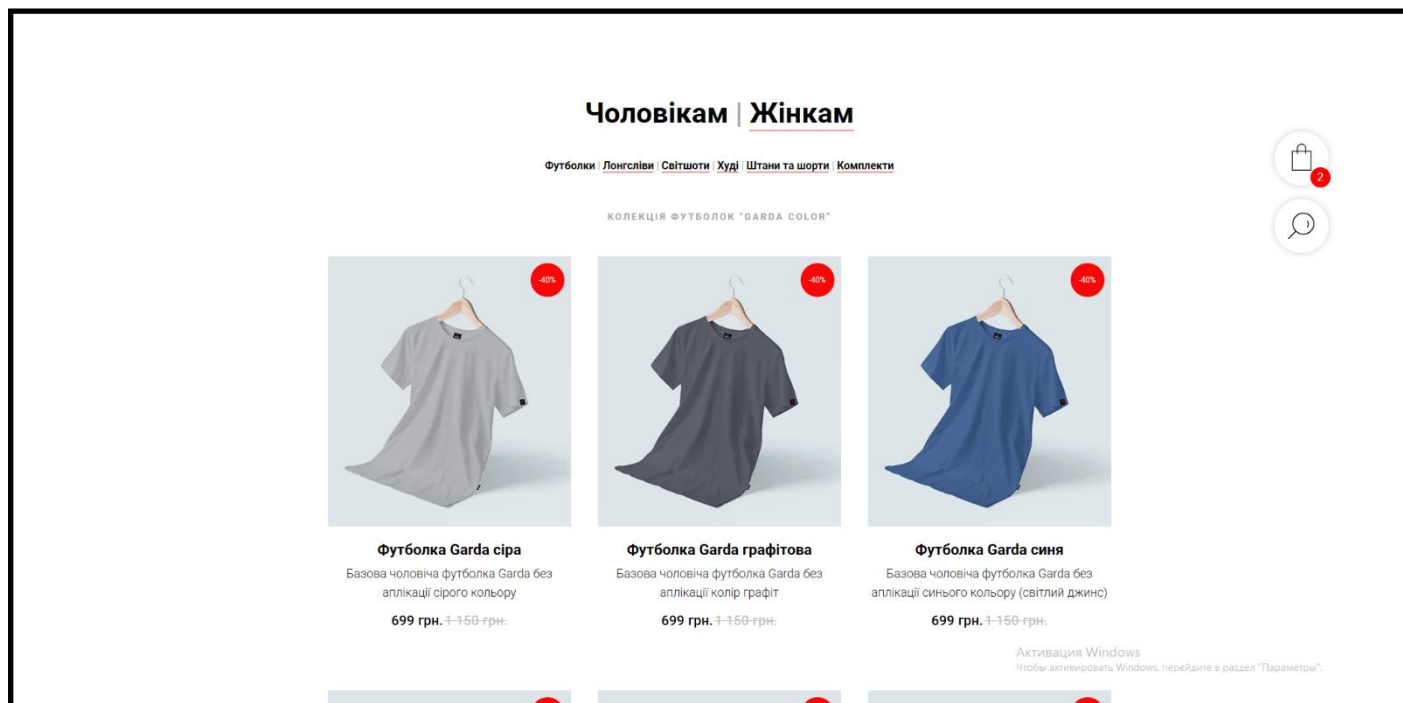


Рис. 1.6 - Сторінка з каталогом товарів інтернет-магазину Garda

Для підведення підсумку було зображено порівняльну таблицю аналогів конкурентів (табл. 1.1).

Таблиця 1.1

Порівняння аналогів конкурентів

Функціонал	Remix	Goldi	Garda	Caverty Mall
Каталог товарів(чоловічий/ жіночий)	+	+	+	+
Розширена фільтрація	+	+	-	+
Пошук товарів	+	+	+	+
Кошик товарів	+	+	+	+
Особистий кабінет користувача	+	+	-	+
Відстеження статусу замовлень	+	-	-	+
Система відгуків на товари	+	+	-	+
Список обраних товарів	-	+	-	-
Сторінка побажань товарів	-	-	-	+

Проаналізувавши існуючі інтернет-магазини одягу можна визначити основні відмінності між ними, одні з них мають широкий функціонал система відгуків: розширений особистий кабінет з відслідковуванням замовлень, у інших лише базовий функціонал каталогу та кошика. Фільтрація товарів у більшості

конкурентів обмежена загальними категоріями без можливості пошуку за розмірами чи ціновим діапазоном.

Враховуючи отриману інформацію при аналізі конкурентів було прийнято рішення розробити власний інтернет-магазин одягу, який усуне проблеми конкурентів.

Таким чином, розроблюваний додаток має забезпечити: зручний та інтуїтивно зрозумілий інтерфейс для покупців; розширену системи фільтрацій товарів за категоріями, розмірами та ціною; особистий кабінет користувача з можливістю відстеження статусу замовлення; систему відгуків на сторінці кожного товару; адміністративну панель для керування товарами, замовленнями та користувачами.

1.3. Вибір архітектури інтернет-магазину одягу

При розробці сучасного вебзастосунку важливо правильно вибрати архітектурний підхід. Він визначає структуру системи, взаємодію компонентів між собою та принципи організації коду [7]. Оскільки неправильний вибір архітектури може привести до проблем з майбутнім масштабуванням системи, погіршення продуктивності та ускладнення в підтримці коду в майбутньому. Саме тому перед початком розробки необхідно проаналізувати існуючі архітектурні підходи та обрати найкращий варіант для поставлених задач.

Однією з таких є монолітна архітектура, яка працює за принципом розробки застосунку як одного єдиного модуля [7]. Клієнтська частина, логіка сервера та бази даних пов'язані між собою та працюють як одне ціле. Всі компоненти працюють в спільному процесі та використовують одну пам'ять.

Вона добре підходить для невеликих проєктів на початку розробки, адже легка у розробці та тестуванні. Таким чином розробнику не потрібно піклуватися про комунікацію між компонентами. Розгортання такого застосунку також є простим, достатнім буде розгорнути лише один артефакт.

Зрештою є і недоліки використання даної архітектури, при масштабуванні проєкту вони можуть виникнути. З розширенням функціональності зростає

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

складність його розуміння та підтримки. Якщо розширювати одну частину додатку доведеться масштабувати весь застосунок, через що масштабування монолітного застосунку є неефективним. Також важливою проблемою є те, що при помилці в одному модулі може відмовити уся система.

Замість монолітної архітектури можна використати мікросервісну архітектуру, яка передбачає розподілення застосунку на невеликі незалежні між собою сервіси, які відповідають за конкретну задачу [5,7]. Наприклад окремі сервіси можуть відповідати за різні задачі, так як: управління товарами, обробку замовлень, авторизацію користувачів чи їх реєстрацію. Їхня взаємодія відбувається через API [13] і можуть розгортатися незалежно один від одного.

Основними перевагами даної архітектури є легка масштабованість, кожний сервіс можна масштабувати незалежно від навантаження [5]. Це дозволяє оновлювати окремі частини системи без зупинки всього застосунку. Також різні сервіси можуть використовувати різні технології, завдяки чому ми можемо мати більшу гнучкість у виборі інструментів.

Основними недоліками є висока складність у розробці та налагодженні порівняно з монолітною архітектурою [5]. Складності також додає необхідність організації міжсервісних комунікацій. Також вона потребує більш складної інфраструктури для розгортання та моніторингу, що в свою чергу збільшує операційні витрати.

Ще одним з можливих варіантів є клієнт-серверна архітектура, яка передбачає розподілення застосунку на дві частини, клієнтську та серверну [7]. Клієнтська відповідає за відображення інтерфейсу користувача та взаємодію з ним. Сервер в свою чергу відповідає за обробку запитів, бізнес-логіку та базу даних. Зв'язок між клієнтом та сервером відбувається за допомогою HTTP протоколу [13].

Клієнтська частина може бути реалізована двома шляхами Multi Page Application (MPA) або Single Page Application (SPA) [11]. У MPA переходи між сторінками призводять до повного перезавантаження сторінки. SPA працює по іншому принципу, він завантажує усе необхідне один раз і оновлює вміст

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

сторінки коли це необхідно, без перезавантаження сторінки, що в свою чергу забезпечує швидкий та зручний користувацький інтерфейс.

Серверна частина реалізує REST API за допомогою якого клієнт отримує та надсилає дані. REST(Representational State Transfer) [6] використовується для побудови програмного інтерфейсу, визначає як буде взаємодіяти між собою клієнт та сервер [13]. Одним з основних принципів є відсутність стану(Stateless), коли запити від клієнта до сервера повинні містити всю необхідну інформацію для обробки. Стан клієнта між запитами при цьому не зберігається на сервері. Також є важливим принцип єдиного інтерфейсу, коли взаємодія між компонентами системи відбувається через HTTP методи GET для отримання даних, POST для створення, PUT для оновлення та DELETE для видалення даних. Для покращення продуктивності використовується кешування, завдяки чому відповіді сервера можуть кешуватися на стороні клієнта. REST API передбачає багаторівневу систему, завдяки якій між клієнтом та сервером можуть міститися різні рівні для розподілення навантаження та кешу [13].

Основними перевагами клієнт-серверної архітектури є розподіл відповідальності між компонентами, незалежна робота серверної та клієнтської частин додатку, можливість легко масштабуватися та підтримувати [7]. Один REST API може використовуватися як для вебзастосунків чи мобільних додатків.

Для порівняння архітектур розробки було наведено таблицю з їх характеристиками (табл. 1.2).

Таблиця 1.2

Порівняння архітектур розробки

Критерій	Монолітна	Мікросервісна	Клієнт-серверна
Складність розробки	Низька	Висока	Середня
Масштабованість	Низька	Висока	Середня
Простота розгортання	Низька	Низька	Середня
Незалежність компонентів	Низька	Висока	Середня
Продуктивність	Висока	Середня	Висока
Підходить для малих проєктів	Так	Ні	Так

Таким чином для розробки інтернет-магазину одягу Caverty Mall було обрано клієнт-серверну архітектуру з REST API [13]. Дана архітектура є оптимальною для розробки даного додатку, адже дозволить забезпечити функціональність системи.

Клієнтська частина додатку реалізована у вигляді Single Page Application в основі якої є бібліотека React [2, 8, 14] з використанням TypeScript [9, 19]. Завдяки SPA забезпечується швидкий перехід між сторінками та без перезавантаження [11]. Серверна частина додатку реалізована на основі Node.js та фреймворку Express [1, 4], завдяки якому забезпечується обробка HTTP запитів та реалізацію бізнес-логіки. Для зберігання даних про товари, користувачів та замовлення використовується база даних Mongo DB [3, 12].

Взаємодія між клієнтом та серверною частиною застосунку відбувається через REST API, дані якого передаються у форматі JSON [13]. Авторизація користувача реалізована за допомогою JWT токена [22], після того як користувач авторизується за ним фіксується токен, який використовується при кожному наступному запиті, щоб підтвердити особу.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки інтернет-магазину одягу було обрано набір сучасних технологій, які добре взаємодіють між собою та дозволяють розробити повноцінний застосунок. Клієнтська частина додатку буде розроблена за допомогою React з TypeScript [8, 19]. Серверна частина побудована на Node.js з використанням технології Express [15]. Для зберігання даних було обрано базу даних MongoDB, де буде зберігатися інформація про клієнтів, товари, замовлення. Для збереження зображень у базі даних буде використано хмарний сервіс Cloudinary.

React - це бібліотека, яка використовується для розробки користувацького інтерфейсу. Головною перевагою є те що інтерфейс додатку розподілений на невеликі незалежні між собою компоненти, завдяки чому ми можемо повторно

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

використовувати їх у будь-яких місцях застосунку, без перезавантаження сторінки.

Для підвищення швидкості роботи застосунку використовується Virtual DOM [8]. Перед тим як одразу вносити зміни в DOM браузера, спочатку відбувається порівняння між старим та новим, і зміни вносять поступово за потреби.

Однією з головних переваг React для розробки даного проєкту є компонентний підхід, завдяки якому ми можемо розбивати інтерфейс додатку на невеликі компоненти, що в свою чергу дозволить спростити та зробити розробку зручнішою [31]. Наприклад картка товару розроблена один раз може використовуватися одразу у декількох місцях на головній сторінці, рекомендованих товарах, каталозі, в блоці схожих товарів та на сторінці товару. Що в свою чергу дозволяє скоротити вміст коду та робить розробку зручнішою. Якщо необхідно внести зміни у компоненті, то це зміниться скрізь де він був використаний.

Є широкий вибір різноманітних бібліотек, що прискорює та полегшує розробку. В проєкті було використано такі бібліотеки як: React Router для здійснення навігації між сторінками [26], Zustand для керування станом застосунку [24], Axios для використання HTTP запитів до сервера [27] Lucide React для іконок проєкту [30].

Зручна робота зі станом застосунку через хуки, у React є вбудовані хуки, які використовуються для керування станом компонентів [14]. Хук useState використовується для збереження локального стану компонента, наприклад обраного розміру товару чи кількості його в кошику [2]. Хук useEffect дозволяє виконувати ефекти, такі як завантаження даних з сервера при відкритті сторінки товару. Також використовується бібліотека Zustand [24], для даних авторизованого користувача та вмісту кошика, вона добре взаємодіє з React та забезпечує зручний доступ до даних з будь-якого компонента.

TypeScript це розширення JavaScript, від Microsoft, яке додає систему типів [9]. Дозволяє вказувати які дані мають бути на вході та виході функцій.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

Головною його перевагою є виявлення помилок ще на етапі розробки застосунку [9]. Наприклад якщо функція очікує число а їй передають рядок, то редактор підкреслить це як помилку. Це значно спрощує розробку, адже у проєкті є багато типів даних товари, замовлення, користувачі а за допомогою TypeScript ми зможемо уникнути велику кількість помилок [19].

Node.js та Express це середовище завдяки якому ми можемо запустити JavaScript не в браузері а на сервері. Особливістю є те, що сервер не чекає поки закінчиться інший запит, щоб почати новий а може виконувати їх паралельно.

Однією з переваг чому було обрано саме Node.js [4], це використання однієї мови JavaScript для фронтенду та для бекенду що в свою чергу спрощує розробку [1]. Також великий каталог пакетів npm дозволяє знайти готове рішення для деяких типових завдань [10].

Express використовується для зручного побудування API [16]. З його використанням не потрібно дотримуватися жорсткої структури проєкту, що в свою чергу дає свободу в розробці коду. Завдяки middleware можна легко додати авторизацію, обробку файлів або логування кількома рядками коду [32].

MongoDB база даних, яка відрізняється від інших реляційних баз даних тим що зберігає дані в документах, а не таблицях [17]. Що в свою чергу дає більшу гнучкість проєкту при розробці, наприклад різні товари можуть мати різні поля без необхідності змінювати всю структуру бази даних[3].

Для інтернет-магазину це особливо зручно, адже кожен товар має свої розміри та характеристики. Наприклад футболка має розмір і колір, а взуття має розмірну сітку. В реляційній базі даних довелося б створювати окремі таблиці або залишити багато полів порожніми, на відміну від MongoDB [12].

Для підключення до MongoDB було використано бібліотеку Mongoose, завдяки якій можна зручно описати структуру даних через схеми та моделі, а також зробити валідацію даних без збереження [18].

Tailwind CSS використовується для стилізації інтерфейсу [20]. Він надає набір класів за допомогою який можна розробити будь-який дизайн, на відміну від готових компонентів.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІЗп	Арк.
		Чижмоторя О.Г.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

Головною перевагою є в тому що дизайн не обмежений готовими шаблонами, а можна налаштувати під свої потреби [20]. Завдяки цьому кінцевий CSS файл містить лише ті стилі які використовуються у проєкті.

Cloudinary хмарний сервіс для зберігання та обробки медіафайлів [21]. Надає програмний інтерфейс завдяки якому можемо завантажувати зображення з застосунку та отримати посилання на нього щоб зберегти в базі даних, та подалі використовувати його.

Для збереження фотографій було обрано саме даний сервіс, оскільки він зберігає зображення на власному сервері, що в свою чергу дозволяє нам не займати дисковий простір та не створює додаткове навантаження на сервер при кожному запиті зображення [21].

Тому завдяки Cloudinary ці проблеми вирішені, зображення завантажуються напряму в хмарне середовище та доставляються користувачу через CDN мережу [21]. CDN(Content Delivery Network) це мережа серверів, яка розташована у різних частинах світу, за допомогою якої контент доставляється з найближчого до користувача сервера, що значно прискорює процес. Сервіс також автоматично оптимізовує розмір та формат зображення, що прискорює завантаження сторінки.

JWT(JSON Web Token) використовується для безпечної передачі даних про користувача між сторінками у вигляді JSON об'єкту [22]. Токен складається з трьох частин: заголовку в якому міститься тип токена та алгоритм шифрування; інформація про користувача; цифровий підпис який забезпечує цілісність токена.

Для авторизації користувачів на сайті було використано JWT токен [22]. Після того як користувач успішно увійде у систему, сформується токен який міститиме ідентифікатор користувача та його роль. Даний токен зберігається на стороні клієнта та передається при кожному новому запиті до сервера.

Коли сервер отримує запит, він перевіряє підпис токена якщо все правильно, то виконується запит відповідний до прав доступу користувача. Якщо токен відсутній або недійсний, тоді сервер поверне помилку 401, що

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмотря О.Г.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

означатиме що користувач не авторизований. Таким чином нам не потрібно зберігати сесію користувача на сервері, що в свою чергу спрощує процес масштабування та зменшує навантаження на базу даних.

Вимоги до апаратного забезпечення

Для розгортання серверної частини застосунку необхідно: процесор з тактовою частотою 1 ГГц; оперативна пам'ять від 512 МБ; вільний дисковий простір від 1 ГБ; підключення до мережі Інтернет.

Для роботи з клієнтською частиною буде достатньо будь-якого сучасного пристрою з доступом в мережу Інтернет.

Висновки до першого розділу

У першому розділі було проаналізовано предметну область та існуючі рішення у сфері онлайн-торгівлі одягом.

В результаті аналізу трьох інтернет-магазинів Remix, Goldi та Garda було виявлено, що більшість з них має обмежений функціонал. Здебільшого більшість конкурентів не мають розширеної системи фільтрації товарів за розмірами та ціновим діапазоном, відсутній особистий кабінет користувача з можливістю відстеження статусу замовлення, також система відгуків або її немає, або реалізована не зручно.

Було обрано клієнт-серверну архітектуру з REST API як оптимальне рішення для розробки інтернет-магазину даного типу. Завдяки такому підходу забезпечиться чіткий поділ між компонентами, та незалежну розробку клієнтської та серверної частин додатку, та масштабованість в майбутньому .

Для розробки системи було обрано сучасний стек технологій: React з TypeScript для клієнтської частини, Node.js та Express для серверної частини, MongoDB як базу даних, Cloudinary для збереження зображень та JWT для авторизації користувачів. Дані технології були обрані таким чином, щоб вони були сумісні між собою та мали високу продуктивність.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ОДЯГУ CAVERTY MALL

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Інтернет-магазин одягу Caverty Mall надає вебінтерфейс завдяки якому, користувачі можуть переглядати каталог товарів, здійснювати пошук, покупки та взаємодію з платформою. Система має велику функціональність, доступ до якої залежить від ролі користувача.

Користувачів системи можна розділити на три основні групи. Незареєстровані користувачі мають доступ лише до перегляду каталогу, сторінки товару та можливості додавання в кошик, для здійснення покупки потрібно авторизуватися в системі [22]. Зареєстровані покупці можуть додавати товари у кошик, здійснювати замовлення, залишати відгуки та відстежувати статус своїх замовлень. Адміністратор має повний доступ до керування системою, товарами, замовленнями, відгуками та користувачами.

Для відображення взаємодії між користувачами та функціональністю системи було реалізовано діаграму варіантів використання системи (рис. 2.1).



Рис. 2.1. Діаграма варіантів використання

Опис діаграми варіантів використання системи наведено в таблицях 2.1-2.8.

Одним з перших, з чим зустрінеться користувач, буде авторизація або реєстрація в системі, завдяки якій старі користувачі можуть авторизуватися, а нові зареєструвати новий акаунт, після чого отримують доступ до повноцінного функціонування системи (табл. 2.1).

Таблиця 2.1

Опис прецеденту «Реєстрація та авторизація»

Назва варіанту використання	Реєстрація та авторизація
Основні актори	Відвідувач, покупець, адміністратор
Короткий опис	Користувачі можуть реєструватися в системі створивши новий обліковий запис або авторизуватися під уже існуючими даними. Після авторизації користувач отримує доступ до персональної інформації та відповідного функціоналу

Наступним є можливість переглядати каталог товарів, завдяки чому користувачі можуть побачити усі наявні товари. Для зручності реалізовано фільтрацію за різними характеристиками (табл. 2.2).

Таблиця 2.2

Опис прецеденту «Перегляд каталогу товарів»

Назва варіанту використання	Перегляд каталогу товарів
Основні актори	Відвідувач, покупець
Короткий опис	Користувачі можуть переглядати каталог товарів магазину, можуть фільтрувати їх за категоріями, розміром та ціновим діапазоном, також сортування та пошук за назвою.

Після того як користувач знайшов те що йому потрібно, він може перейти на сторінку товару. На сторінці можна дізнатися детальну характеристику, наявні розміри, фотографію та відгуки інших користувачів (табл. 2.3).

Таблиця 2.3

Опис прецеденту «Перегляд сторінки товару»

Назва варіанту використання	Перегляд сторінки товару
Основні актори	Відвідувач, покупець
Короткий опис	Користувачі можуть переглядати детальну інформацію про товар, фотографію, опис, характеристики, можливі

	розміри з кількістю їх залишку, рейтинг та відгуки покупців.
--	--

Після перегляду детальної інформації про товар, користувач може додати його у кошик [24]. Також у кошику є можливість змінювати кількість товарів та перейти до оформлення замовлення (табл. 2.4).

Таблиця 2.4

Опис прецеденту «Керування кошиком»

Назва варіанту використання	Керування кошиком
Основні актори	Покупець
Короткий опис	Зареєстровані користувачі можуть додавати товари до кошика обравши потрібний розмір та кількість, у кошику можна змінити кількість товарів або видалити їх.

Коли користувач обрав все необхідне, він може перейти до оформлення замовлення. Відобразиться форма, де необхідно буде вказати персональні дані та обрати спосіб доставки (табл. 2.5).

Таблиця 2.5

Опис прецеденту «Оформлення замовлень»

Назва варіанту використання	Оформлення замовлень
Основні актори	Покупець
Короткий опис	Покупець може оформляти замовлення вказавши дані отримувача, обравши спосіб доставки через Нову Пошту або Укрпоштою з пошуком міста та відділення через API Нової Пошти [23]

Після того як користувач оформив замовлення, є можливість переглянути його статус в особистому кабінеті. Також є можливість відмінити замовлення за потреби (табл. 2.6).

Таблиця 2.6

Опис прецеденту «Керування замовленнями»

Назва варіанту використання	Керування замовленнями
Основні актори	Покупець, Адміністратор
Короткий опис	Покупець може переглядати історію своїх замовлень та скасовувати замовлення за потреби. Адміністратор має доступ до всіх замовлень системи з можливістю зміни їх статусу.

Користувачі можуть залишати відгуки до товарів, вказується рейтинг та текст. Також можна відповідати на відгуки інших користувачів. Адміністратор в свою чергу може видаляти відгуки, якщо він порушує правила платформи (табл. 2.7).

Таблиця 2.7

Опис прецеденту «Система відгуків»

Назва варіанту використання	Система відгуків
Основні актори	Покупець, Адміністратор
Короткий опис	Покупець може залишати відгуки з рейтингом на сторінці товару, відповідати на відгуки інших користувачів, редагувати та видаляти власні відгуки. Адміністратор має право видаляти будь-які відгуки.

Для адміністратора є необхідна можливість додавати нові товари, редагувати старі та переглядати список замовлень. При додаванні товарів можна додати зображення та характеристики товару (табл. 2.8).

Таблиця 2.8

Опис прецеденту «Керування товарами»

Назва варіанту використання	Керування товарами
Основні актори	Адміністратор
Короткий опис	Адміністратор може додавати нові товари з фотографіями через сервіс Cloudinary [21], редагувати існуючі товари змінюючи їх характеристики та кількість по розмірах, а також видаляти товари з каталогу.

Розділ «Список бажань», завдяки якому користувачі можуть додавати побажання товарів, які вони хотіли б побачити в майбутньому асортименті магазину. Після розгляду пропозиції адміністратором буде відображено помітку зі статусом розгляду (табл. 2.9).

Таблиця 2.9

Опис прецеденту «Список бажань»

Назва варіанту використання	Список бажань
Основні актори	Покупець, адміністратор
Короткий опис	Покупець може додавати побажання щодо товарів які він хотів би бачити в майбутньому в магазині вказуючи назву, категорію, бажану ціну та фото для прикладу. Адміністратор переглядає побажання та змінює їх статус.

Основна ціль інтернет-магазину одягу Caverty Mall полягає у створенні зручної платформи, за допомогою якої покупці можуть знаходити та замовляти одяг не виходячи з дому. Платформа спрямована на забезпечення комфортної взаємодії між продавцем та клієнтом, від перегляду каталогу до отримання замовлення.

За допомогою магазину покупці мають можливість переглядати широкий асортимент товарів з детальним описом та фотографіями, обрати потрібний розмір, прочитати відгуки інших покупців та оформити замовлення.

Основною метою розробленої платформи є створення повноцінного інструменту для онлайн-торгівлі одягом, який забезпечує зручне здійснення покупок клієнтом та керування асортиментом і замовленнями для адміністратора магазину.

Вимоги користувачів наведено в табл. 2.10.

Таблиця 2.10

Вимоги користувачів

Внутрішній користувач – адміністратор	В адміністратора повинна бути можливість додавати нові товари через платформу з описом, фотографіями, кількістю та розмірами. Редагувати уже існуючі товари, їх опис та наявні розміри включаючи кількість кожного з них окремо. Також можливість видалення товарів, які застаріли чи більше не будуть в наявності. Необхідна можливість перегляду та обробки замовлень покупців, щоб адміністратор міг бачити всі зареєстровані замовлення у системі з детальною інформацією про товари, дані доставки та покупця. Також можливість змінювати статус замовлення від "Очікує" до "Доставлено", що в свою чергу буде відображатися в особистому кабінеті покупця. Керувати користувачами системи з можливістю переглядати усіх зареєстрованих користувачів, пошук їх та зміна ролі. Опрацьовувати побажання користувачів щодо нових товарів, адміністратор повинен бачити усі побажання покупців які вони додали у розділі "Список бажань". Також можливість видалення відгуків користувачів які порушують правила магазину.
Внутрішній користувач – покупець	Реєстрація та авторизація користувачів у системі [22], щоб користувачі могли створювати новий обліковий запис вказавши своє ім'я, email та пароль або увійти за уже існуючим обліковим записом, якщо він був створений раніше. Перегляд каталогу товарів, фільтрація та пошук, який включає в себе перегляд усіх товарів магазину з можливістю їх фільтрації за категоріями, розміром та ціновим діапазоном. Також можливість

	відсортувати відображення товарів за ціною, популярністю та датою створення. Додавання товарів до кошика та оформлення замовлень [24], після того як користувач знайшов необхідний йому товар він може додати його до кошика, після чого або продовжити вибір товарів далі або оформити замовлення. Вибір способу доставки товару при оформленні замовлення покупець, щоб обрати зручний спосіб доставки з двох можливих через "Нову Пошту" або "Укрпошту" [23]. Після чого вводиться назва міста і система пропонує доступні відділення через API Нової Пошти. Перегляд історії замовлень та відстеження їх статусу, завдяки якому зареєстрований користувач в особистому кабінеті бачить історію своїх замовлень та статус ще незавершених. Скасування замовлень, щоб користувач міг скасувати своє замовлення на будь-якому етапі. Залишення відгуків та рейтинг товарів, для того щоб покупець міг залишити відгук на товар у текстовому вигляді, а також оцінку від 1 до 5. Також є можливість відповідати на коментарі інших користувачів, редагувати та видаляти власний відгук. Додавання бажаних товарів у розділі "Список бажань", користувачі системи можуть пропонувати товари, які вони хотіли б бачити у магазині. При цьому вказується назва товару, категорія, бажана ціна та завантажити фото для прикладу. Редагування власного профілю, кожен користувач має можливість змінювати власну інформацію в особистому кабінеті.
Внутрішній користувач – відвідувач	Незареєстрований користувач може переглядати увесь асортимент магазину. Також має можливість переглядати детальну інформацію про товар його фотографії, опис, розміри та відгуки інших покупців. Здійснювати пошук за назвою та фільтрація за категоріями, розміром та ціною. Відвідувач може додавати товари до кошика, однак для здійснення замовлення йому необхідно буде зареєструватися в системі.

Для повноцінної працездатності системи було визначено функціональні та нефункціональні вимоги. Функціональні вимоги описують дії системи та наданий функціонал користувачам з різними ролями. Функціональні вимоги наведено в табл. 2.11.

Таблиця 2.11

Функціональні вимоги до системи

№	Вимоги
1	Реєстрація та авторизація користувачів з різними правами доступу, а саме відвідувач, покупець, адміністратор.
2	Забезпечити зручне використання пошуку та фільтрації товарів за категоріями, розміром та ціновим діапазоном, з використанням декількох фільтрів одночасно.

3	Реалізувати процес оформлення замовлень з автоматичним пошуком міста та відділення через API Нової Пошти.
4	Реалізувати можливість залишати відгуки та рейтинг на сторінках товарів, з можливістю редагування та видалення їх за потреби, а також можливість відповідати на коментарі інших користувачів.
5	Реалізувати можливість відстеження статусу власних замовлення та можливість скасувати їх у особистому кабінеті.
6	Реалізувати панель для адміністратора за допомогою якої здійснюється керування товарами, замовленнями, користувачами та побажаннями покупців.

Нефункціональні вимоги описують продуктивність, надійність, безпеку та зручність використання системи. Вони є важливими для забезпечення стабільного використання платформи. Нефункціональні вимоги наведено в табл. 2.12.

Таблиця 2.12

Нефункціональні вимоги до системи

№	Вимоги
1	Легке освоєння інтерфейсу, яке не повинно перевищувати 5 хвилин для нового користувача завдяки інтуїтивному розташуванню елементів.
2	Час відповіді системи на дії користувача не повинен перевищувати 2 секунди при великому навантаженні.
3	Інтерфейс системи повинен бути зручним, інтуїтивно зрозумілим та не вимагати додаткових знань від користувача.
4	Система повинна бути надійною та коректно обробляти усі помилки та повідомляти лише зрозумілими повідомленнями без відображення технічних помилок
5	Дані користувачів, товарів, замовлень повинні надійно зберігатися у базі даних MongoDB [17].
6	Платформа повинна працювати цілодобово.
7	Паролі користувачів повинні зберігатися лише у зашифрованому вигляді [28].
8	Авторизація користувачів через JWT токени [22], термін якого 7 днів після чого користувачу потрібно авторизуватися повторно.

Основними системними вимогами до клієнтського пристрою є: наявність оперативної пам'яті не менше 512 МБ; процесор з тактовою частотою не нижче 1 ГГц; стабільний інтернет; встановлений сучасний браузер, такий як Google Chrome, Mozilla Firefox або Microsoft Edge. Дані вимоги є мінімальними для більшості сучасних пристроїв тому платформа є легкодоступною, та не вимагає додаткового обладнання.

Основними вимогами до серверної частини є: середовище виконання Node.js не нижче 18 [15]; база даних MongoDB з підключенням через сервіс MongoDB Atlas [17]; підключення до хмарного сервісу Cloudinary для збереження зображень до товарів [21]. Такий метод збереження зображень допоможе зменшити навантаження на базу даних, особливо при великій кількості користувачів.

Аналіз функціональних вимог дозволив визначитися з основними сутностями, які забезпечать програмну реалізацію інтернет-магазину одягу Saverty Mall. Було зображено діаграму класів яка використовується для роботи додатку (рис. 2.2).

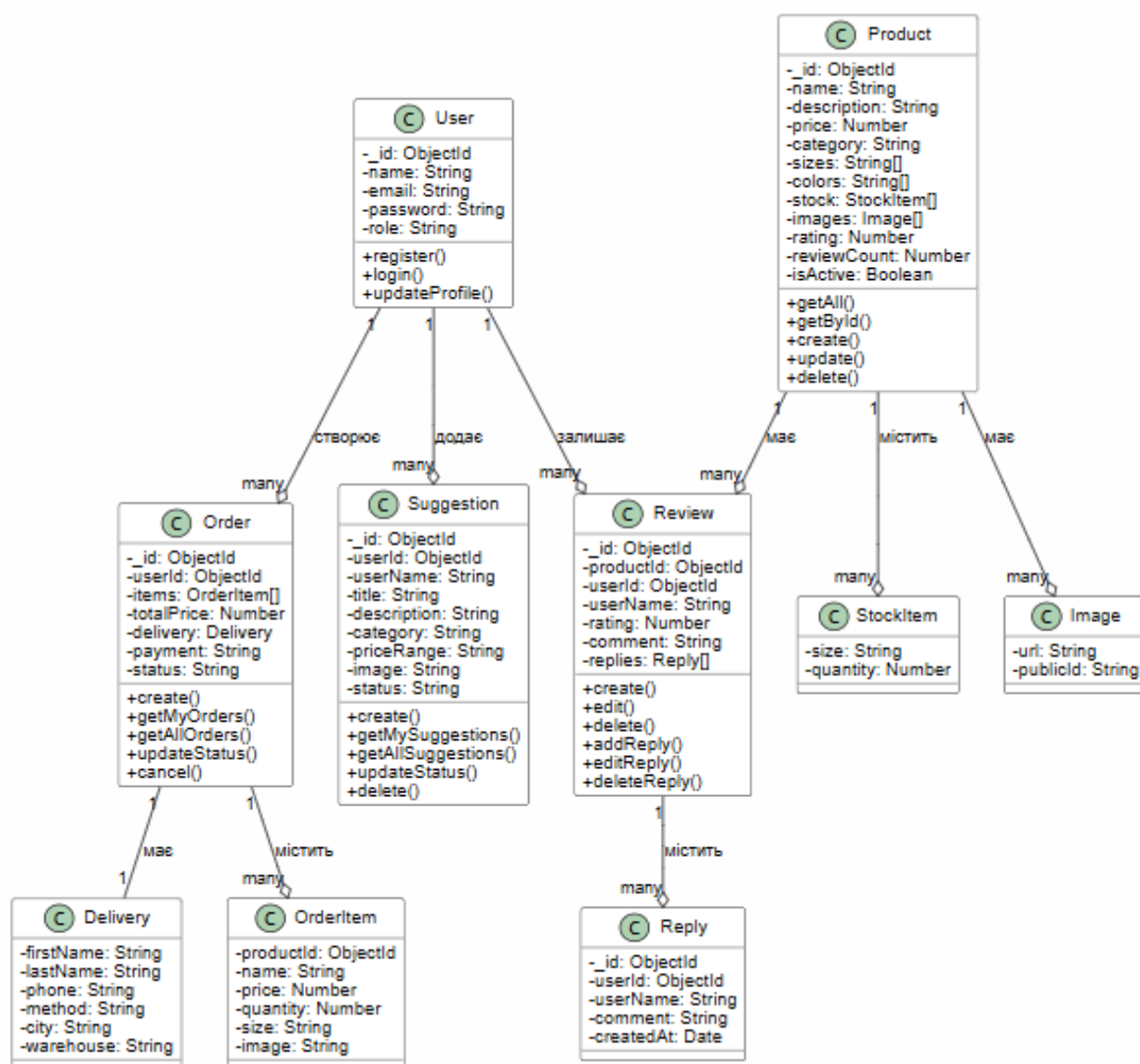


Рис. 2.2. Діаграма класів

Клас User відповідає за користувача системи, та містить методи для взаємодії з ним. Register() відповідає за реєстрацію нового користувача та хешуванням пароллю [28]. Login() відповідає за авторизацію користувачів з генерацією JWT токена [22]. UpdateProfile() відповідає за оновлення особистих даних користувача.

Клас Product відповідає за товари магазину. GetAll() отримання списку товарів з фільтрацією та сортуванням. GetById() отримання інформації про конкретний товар за ID. Create() створення нового товару адміністратором з завантаженням фото через Cloudinary. Update() редагування опису, кількості та розміри уже існуючого товару. Delete() видалення товару з каталогу.

Допоміжний клас StockItem використовується для збереження інформації про залишок товару по кожному розміру окремо, містить поля розмір та кількість.

Клас Order відповідає за замовлення покупців. Create() відповідає за створення нового замовлення з даними доставки та списком товарів. GetMyOrders() використовується для того щоб користувач міг отримати дані про всі його замовлення. GetAllOrder() використовується для адміністратора, з метою відображення усіх активних замовлень в системі. UpdateStatus() використовується для зміни статусу замовлення адміністратором. Cancel() скасування замовлення клієнтом.

Клас OrderItem відповідає за збереження інформації про конкретний товар у замовленні вказуючи його назву, ціну, кількість, розмір та фото. Клас Delivery використовується для збереження даних доставки замовлення ім'я та прізвище отримувача, телефон, спосіб доставки, місто та відділення [23].

Клас Review використовується для реалізації відгуків на товари. Create() створення відгуку в текстовому вигляді з рейтингом від 1 до 5. Edit() відповідає за редагування власного відгуку. Delete() використовується для видалення відгуку користувачем який його написав або адміністратор у разі порушення правил платформи. AddReply() використовується для того щоб користувач міг відповісти на коментар іншого. EditReply() використовується для редагування

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмотря О.Г.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

власних відповідей. DeleteReply() використовується для видалення відповіді, може видалити як автор так і адміністратор. Клас Reply використовується для збереження відповідей користувачів на відгуки.

Клас Suggestion відповідає за побажання клієнтів. Та в подальшому відображення в адміністративній панелі, для подальшої взаємодії з ним. Create() додання нового бажаного товару в магазині з можливістю завантаження фото. GetMySuggestions() використовується щоб відобразити власні побажання користувача. GetAllSuggestions() використовується для відображення усіх побажань для адміністратора. UpdateStatus() використовується для зміни статусу побажання. Delete() видалення побажання.

2.2. Розробка бази даних системи

Важливу роль у функціонуванні інтернет-магазину відіграє база даних, оскільки вона забезпечує збереження інформації та швидкий доступ до неї. Саме тому для збереження даних було обрано MongoDB [12].

База даних проєкту складається з п'яти колекцій, а саме: users, products, orders, reviews та suggestions. Фізична схема бази даних зображена на рис. 2.3.

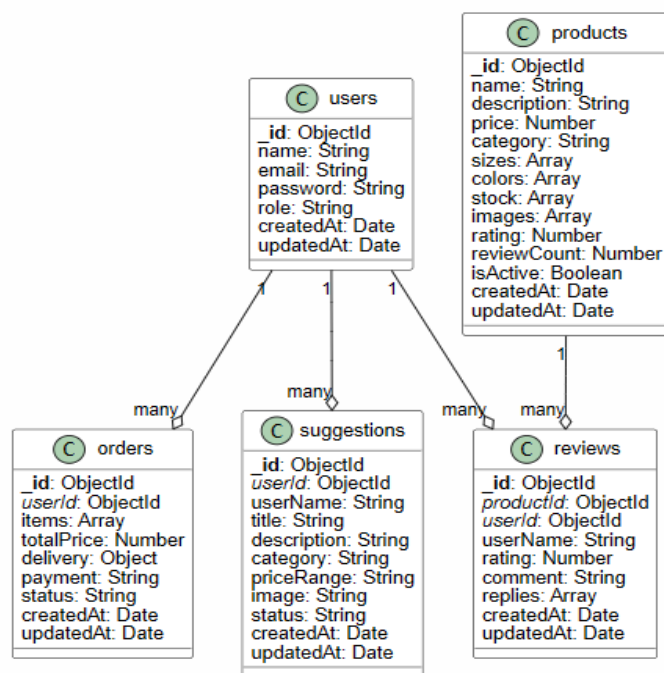


Рис. 2.3 Фізична схема бази даних інтернет-магазину Caverty Mall

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О. Г.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

Колекція User призначена для збереження інформації про зареєстрованих користувачів системи. Кожен користувач має унікальний ідентифікатор, ім'я, електронну пошту, захешований пароль, дату створення та роль користувача (табл. 2.13).

Таблиця 2.13

Структура колекції users

Поле	Тип	Обов'язкове	Опис
Id	Objectid	Так	Унікальний ідентифікатор користувача
Name	String	Так	Ім'я користувача
Email	String	Так	Унікальна електронна пошта
Password	String	Так	Захешований пароль [28]
role	String	Так	Роль користувача
createdAt	Date	Автоматично	Дата реєстрації
updatedAt	Date	Автоматично	Дата останнього оновлення

Колекція Product відповідає за збереження інформації про всі товари магазину з детальною інформацією [17]. Завдяки полям price, category, sizes, colors, stock є можливість здійснювати розширену фільтрацію товарів (табл. 2.14).

Таблиця 2.14

Структура колекції product

Поле	Тип	Обов'язкове	Опис
Id	objectID	Так	Унікальний ідентифікатор для кожного продукту
Name	String	Так	Назва товару
Description	String	Так	Опис товару
Price	Number	Так	Ціна товару в гривнях
Category	String	Так	Категорія товару
Sizes	Array	Ні	Масив доступних розмірів
Colors	Array	Ні	Масив доступних кольорів
Stock	Array	Ні	Масив який відповідає за те, щоб у кожного розміру товару була визначена кількість
Images	Array	Ні	Масив у якому зберігаються посилання на зображення до продукту
rating	Number	Ні	Середній рейтинг товару
reviewCount	Number	Ні	Кількість відгуків
isActive	Boolean	Так	Чи активний товар
createdAt	Date	Автоматично	Дата створення

Колекція Orders відповідає за збереження інформації про усі замовлення покупців. Поле "item" є масивом об'єктів який містить інформацію про кожен товар у замовленні (табл. 2.15).

Таблиця 2.15

Структура колекції orders

Поле	Тип	Обов'язкове	Опис
Id	objectID	Так	Унікальний ідентифікатор для кожного замовлення
userID	objectID	Так	Посилання на користувача
Item	Array	Так	Перелік товарів які було замовлено
totalPrice	Number	Так	Загальна сума замовлення
Delivery	Object	Так	Дані доставки
Payment	String	Так	Спосіб оплати
Status	String	Так	Статус замовлення
createdAt	date	Автоматично	Дата створення

Поле "delivery" є вкладеним об'єктом з даними доставки. Відповідає за збереження персональних даних отримувача та метод доставки (табл. 2.16).

Таблиця 2.16

Структура колекції delivery

Поле	Тип	Обов'язкове	Опис
firstName	String	Так	Ім'я отримувача
lastName	String	Так	Прізвище отримувача
Phone	String	Так	Номер телефону
method	String	Так	Спосіб доставки
City	String	Так	Місто доставки
warehouse	String	Так	Відділення пошти [23]

Колекція Reviews відповідає за збереження відгуків користувачів. Має посилання на товар, щоб відображатися на сторінці та прив'язку до користувача, для можливості редагувати відгук (табл. 2.17).

Таблиця 2.17

Структура колекції reviews

Поле	Тип	Обов'язкове	Опис
Id	objectID	Так	Унікальний ідентифікатор для кожного відгуку
productId	objectID	Так	Посилання на товар
userID	objectID	Так	Посилання на користувача, який залишив цей відгук

userName	string	Так	Ім'я автора відгуку
Rating	Number	Так	Рейтинг до товару від 1 до 5
Comment	String	Ні	Текст відгуку
Replies	Array	Ні	Масив з відповідями на відгук
createdAt	Date	Автоматично	Дата створення

Колекція Suggestions зберігає побажання покупців щодо нових товарів які вони хотіли б бачити в асортименті магазину. Після чого відразу відображається в панелі адміністратора (табл. 2.18).

Таблиця 2.18

Структура колекції suggestions

Поле	Тип	Обов'язкове	Опис
Id	objectID	Так	Унікальний ідентифікатор для кожного запропонованого товару
userID	objectID	Так	Посилання на користувача
userName	String	Так	Ім'я користувача, який пропонує товар
Title	String	Так	Назва бажаного товару
Description	String	Ні	Детальний опис
Category	String	Ні	Категорія товару
priceRange	String	Ні	Бажаний ціновий діапазон
image	String	Ні	Посилання на фото прикладу
Status	String	Так	Статус який отримав бажаний товар після опрацювання адміністратором
createdAt	Date	автоматично	Дата створення

Всі дані розробленого інтернет-магазину зберігаються в хмарній базі даних MongoDB. На відміну від локальних баз даних, хмарна надає постійний доступ до даних незалежно від локального сервера.

Всі операції з даними виконуються за допомогою REST API [6]. Кожен запит перевіряється на наявність та валідність JWT токenu [22], після чого система надає доступ користувачеві до того функціоналу, який відповідає його ролі.

Після створення кожного нового запису в базі даних товару, замовлення, відгуку чи реєстрація користувача система присвоює унікальний ідентифікатор та зберігає дату створення. Завдяки цьому є можливість відстеження усіх змін в

системі. Для взаємодії з базою даних використовується бібліотека Mongoose, яка забезпечує зручний інтерфейс для читання, створення, оновлення та видалення даних. Перед кожним збереженням даних здійснюється валідація, завдяки чому некоректні дані не будуть записані у базу даних.

2.3. Проєктування та реалізація алгоритмів роботи системи

Авторизація користувача в системі складається з наступних дій. Користувач вводить власну електронну пошту та пароль у форму для авторизації. Після чого натискає кнопку "Увійти" дані передаються на сервер через POST запит [13]. Сервер здійснює пошук даного користувача в базі даних за електронною поштою [17]. Якщо користувача не знайдено, сервер повертає помилку 401 і користувач бачить повідомлення що дані введенні не вірно. Якщо користувача знайдено відбувається перевірка введеного паролю та захешованого, який збережений у базі даних [28]. Якщо він не правильний повертається помилка 401. Коли всі дані введенні правильно та сервер знайшов їх, даному користувачеві призначається JWT токен [22].

Діаграма алгоритму авторизації представлена на рис. 2.4.

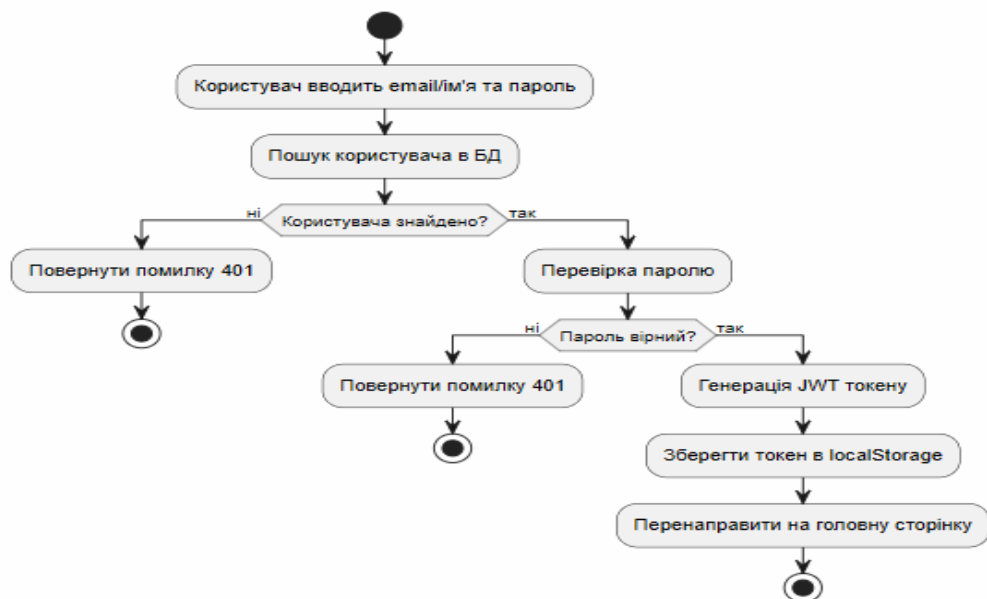


Рис. 2.4. Діаграма алгоритму авторизації

Токен та дані користувача повертається до користувацького інтерфейсу, передається ім'я, електронна пошта та роль. На стороні клієнта токен зберігається у localStorage та використовується для всіх наступних запитів до сервера впродовж 7 наступних днів [22].

Процес оформлення замовлення починається зі сторінки кошику [24]. Спочатку система перевіряє чи користувач авторизований, якщо ні відбувається перенаправлення на сторінку для входу в систему. Якщо кошик порожній повідомляється про відсутність товарів. Якщо ж товари додані та користувач авторизований, він переходить на сторінку з заповненням даних доставки, а саме: ім'я та прізвище, номер телефону, місто та відділення пошти [23]. Після заповнення форми покупець натискає кнопку "Оформити замовлення". Відбувається перевірка чи всі поля заповнені, якщо ні відображається повідомлення про помилку. Якщо всі дані заповнені коректно тоді дані передаються на сервер та створюється замовлення в базі даних зі статусом "Очікує". Після чого користувач бачить повідомлення про успішне оформлення замовлення.

Діаграма алгоритму оформлення замовлення представлена на рис. 2.5.

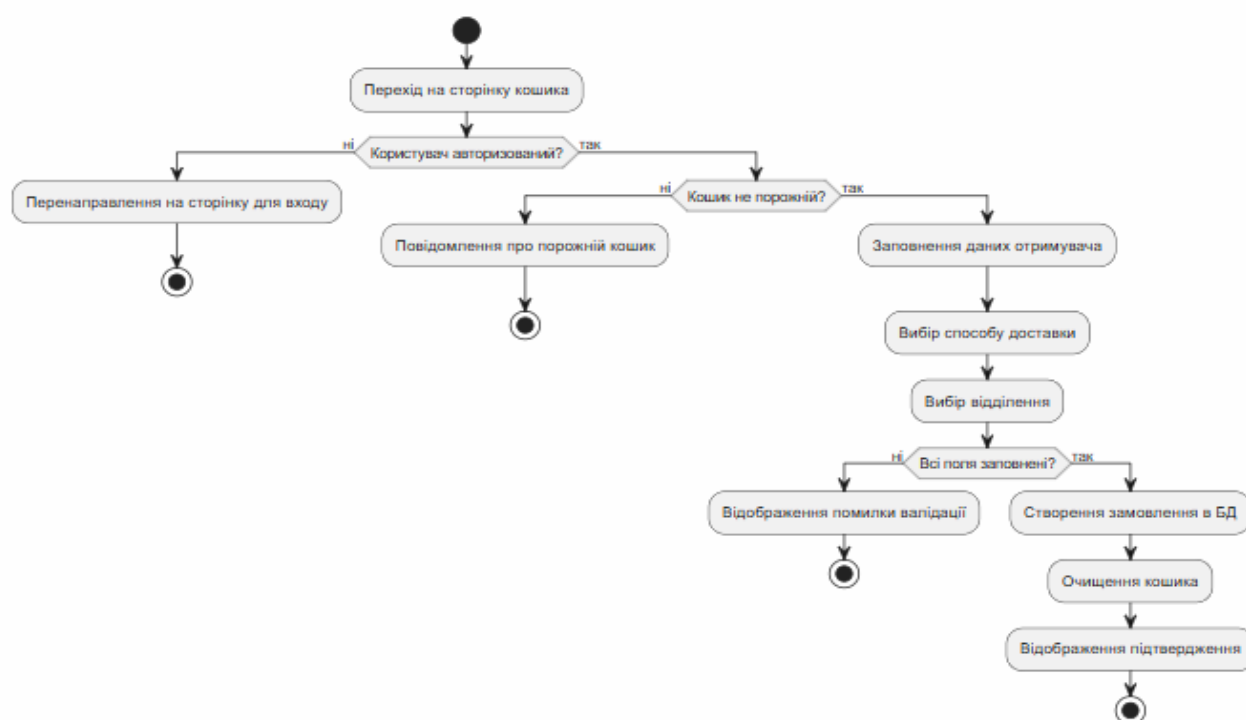


Рис. 2.5. Діаграма алгоритму оформлення замовлення

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмотря О.Г.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

Додавання нового товару доступно лише адміністратору системи. Адміністратор відкриває адміністративну панель та переходить до вкладки "Товари". Натискає кнопку "Додати товар" після чого відкривається форма. Після чого необхідно заповнити обов'язкові поля, а саме : назву, опис, ціну, категорію та фото. Також необхідно обрати розміри товару та вказати їх кількість для кожного розміру окремо. Система перевіряє чи всі поля заповнені, відбудеться валідація [18], якщо щось не правильно система про це повідомить. Якщо всі дані вказані правильно створиться об'єкт FormData з даними товару та зображенням [29]. Потім дані передаються на сервер через POST [6], відбувається перевірка чи має користувач роль адміністратора [22], зображення завантажується на Cloudinary та повертається URL посилання, яке і зберігається у базі даних [21]. Доданий товар зберігається у базі даних та його можна бачити на сторінці з товарами.

Діаграма алгоритму додавання товарів представлена на рис. 2.6.

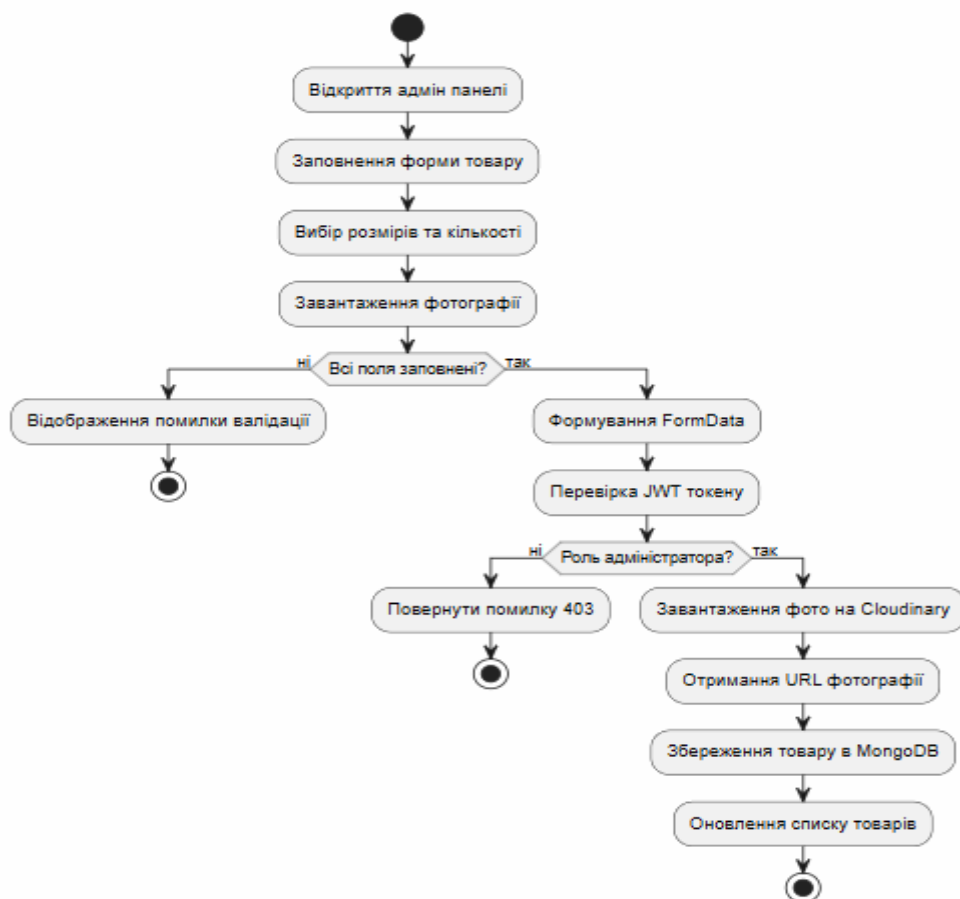


Рис. 2.6. Діаграма алгоритму додавання товарів

Фільтрація товарів відбувається на сторінці каталогу. Користувач обирає параметри фільтрації які йому необхідні або вводить в пошук. Після чого відбувається пошук по всім продуктам, які відповідають даним параметрам. Сервер отримує параметри та робить запит до бази даних з потрібними умовами. Для фільтрації за розмірами використовується масив "stock", який містить розміри та кількість таких товарів в наявності. Після того як потрібні товари були знайдені, вони передаються на сервер та виводяться користувачу.

Діаграма алгоритму фільтрації товарів представлена на рис. 2.7.

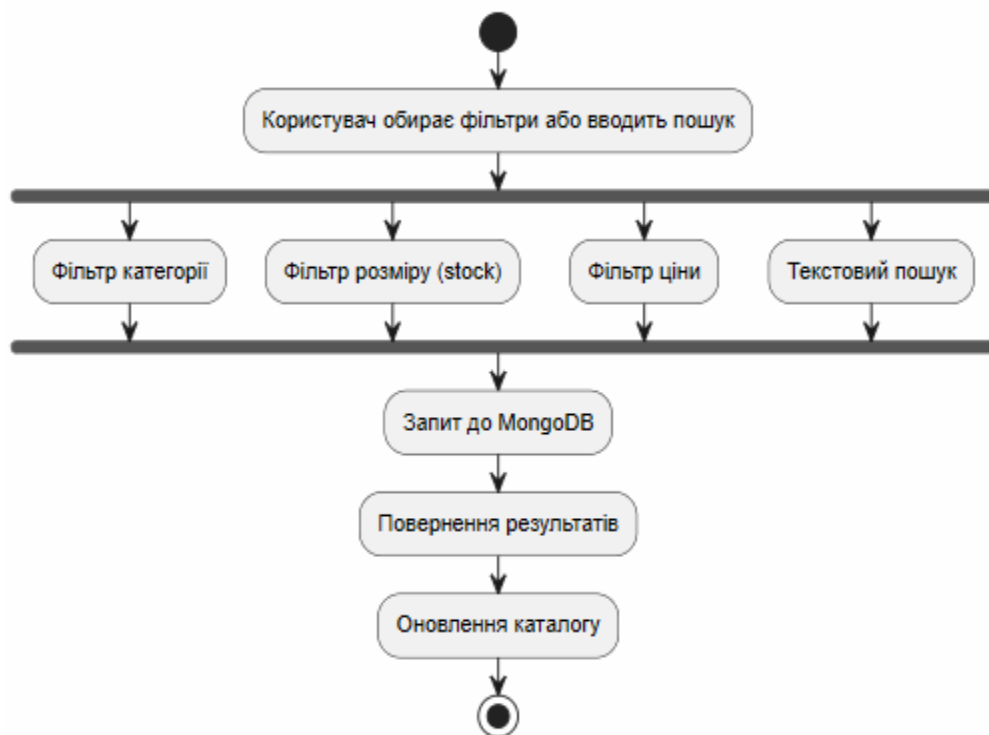


Рис. 2.7. Діаграма алгоритму фільтрації товарів

2.4. Реалізація вебзастосунку інтернет-магазину одягу

Для здійснення ключових операцій на платформі необхідно авторизуватися в системі. Саме тому система авторизації є одним з головних елементів безпеки інтернет-магазину. Завдяки їй користувачі з різними ролями мають свій функціонал. Авторизація користувачів базується на використанні JWT токенів, який призначається кожному авторизованому користувачу та

використовується для перевірки при кожному запиті до захищеного функціоналу.

Для того щоб звичайний користувач не зміг користуватися функціоналом який був призначений для адміністратора було розроблено два middleware [32], а саме "protect" та "adminOnly". Функція "protect" необхідна для перевірки чи є актуальним JWT токен [22]. Якщо він відсутній, прострочений чи недійсний система виведе помилку 401 і запит буде не виконано.

```
export const protect = async (req, res, next) => {
  const token = req.headers.authorization?.split(' ')[1]
  if (!token) return res.status(401).json({ message: 'Not authorized' })
  const decoded = jwt.verify(token, process.env.JWT_SECRET)
  const user = await User.findById(decoded.id)
  if (!user) return res.status(401).json({ message: 'User not found' })
  req.user = { id: user._id.toString(), role: user.role }
  next()}

```

Функція "adminOnly" необхідна для перевірки ролі користувача, якщо звичайний користувач спробує здійснити дію, яка призначена для адміністратора система надішле помилку 403 [32].

```
export const adminOnly = (req, res, next) => {
  if (req.user?.role !== 'admin') {
    return res.status(403).json({ message: 'Admin access required' })
  }
  next()
}
```

Для інтернет-магазину також є важливим фільтрація товарів. Покупець може застосовувати одразу декілька фільтрів. Саме тому функція "getProducts" відповідає за фільтрацію товарів за категорією, розміром, ціновим діапазоном та пошуком за назвою. Для того щоб створити фільтрацію за розмірами, було створено масив "stock" який містить розміри та кількість товарів конкретного розміру.

```
export const getProducts = async (req: Request, res: Response
): Promise<void> => {try {
  const {
    category, size, sort,
    search, minPrice, maxPrice,
    page = 1, limit = 10
  } = req.query
  const filter: any = { isActive: true }
  if (category) filter.category = category
  if (search) filter.$text = { $search: search as string }
  if (minPrice || maxPrice) {

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    filter.price = {}
    if (minPrice) filter.price.$gte = Number(minPrice)
    if (maxPrice) filter.price.$lte = Number(maxPrice)
  } if (size) {
    filter.stock = {
      $elemMatch: { size: size, quantity: { $gt: 0 } } }
  }
  const products = await Product.find(filter)
    .sort(sortOption)
    .skip((Number(page) - 1) * Number(limit))
    .limit(Number(limit))
  res.json({ success: true, data: products }) } catch {
  res.status(500).json({ message: 'Server error' })}}

```

Для збереження зображень товарів було використано хмарне середовище Cloudinary. При додаванні нового товару на сайт адміністратор додає зображення у відповідну для цього форму. Після чого дані відправляються на сервер, де здійснюється обробка файлів під час якого файл з зображенням передається до сервісу Cloudinary, а у відповідь отримує URL посилання на збережене зображення, це посилання і зберігається у базі даних [21].

```

const storage = new CloudinaryStorage({
  cloudinary,
  params: {
    folder: 'caverty-mall',
    allowed_formats: ['jpg', 'jpeg', 'png', 'webp'],
    transformation: [{
      width: 800,
      height: 1000,
      crop: 'limit'
    }],
  },
  as any,})
export const upload = multer({ storage })

```

Важливим елементом системи є можливість залишати відгуки з рейтингом. Тому кожний зареєстрований користувач має можливість залишати відгуки. Після кожного нового відгуку система перераховує середній рейтинг товару та оновлює його в базі даних.

```

const review = await Review.create({
  productId, userId: req.user?.id,
  userName: req.body.userName,
  rating: req.body.rating,
  comment: req.body.comment,
})
const reviews = await Review.find({ productId })
const avgRating = reviews.reduce(
  (sum, r) => sum + r.rating, 0) / reviews.length
await Product.findByIdAndUpdate(productId, {

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    rating: Math.round(avgRating * 10) / 10,
    reviewCount: reviews.length,
  })

```

Для того щоб оформлювати замовлення стало зручніше було реалізовано інтеграцію з API Нової Пошти [23]. Після того як користувач вводить назву міста система автоматично запропонує доступні відділення.

Запити до API Нової Пошти виконуються з клієнтської частини додатку методом POST, для пошуку міста використовується метод "searchSettlements", а для отримання відділення метод "getWarehouses" [23]. Завдяки чому оформлення замовлення для покупця стає зручнішим, та зменшує ризики помилки при відправці товару.

```

const res = await fetch('https://api.novaposhta.ua/v2.0/json/', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    apiKey: NP_API_KEY,
    modelName: 'Address',
    calledMethod: 'searchSettlements',
    methodProperties: { CityName: query, Limit: 10 },
  }),
})
const data = await res.json()
setCities(data.data?.[0]?.Addresses || [])

```

Для зручного здійснення покупок на сайті було розроблено кошик, який містить масив товарів а також обчислює загальну суму всього що до нього додано [24]. При додаванні товару система перевіряє чи доданий уже такий товар, якщо так тоді просто збільшується кількість. Вміст кошика зберігається в локальному сховищі браузера, та залишається там до тих пір поки користувач не вийде повністю з системи.

```

export const useCartStore = create<CartState>()(
  persist(
    (set, get) => ({
      items: [],
      addItem: (product, size, quantity = 1) => {
        const existing = get().items.find(
          (i) => i.product._id === product._id && i.size === size
        )

        if (existing) {

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				43
Змн.	Арк.	№ докум.	Підпис	Дата		

```

set((state) => ({
  items: state.items.map((i) =>
    i.product._id === product._id && i.size === size
    ? { ...i, quantity: i.quantity + quantity }
    : i ),,))) else {
set((state) => ({
  items: [...state.items, { product, size, quantity }],,))),clearCart:
() => set({ items: [] })),
{ name: 'cart-storage' })))

```

Для підведення підсумку було створено діаграму компонентів інтернет-магазину Caverty Mall. На діаграмі відображена взаємодія між модулями клієнтської та серверної частин додатку, а також інтеграція з зовнішніми сервісами (рис. 2.8).

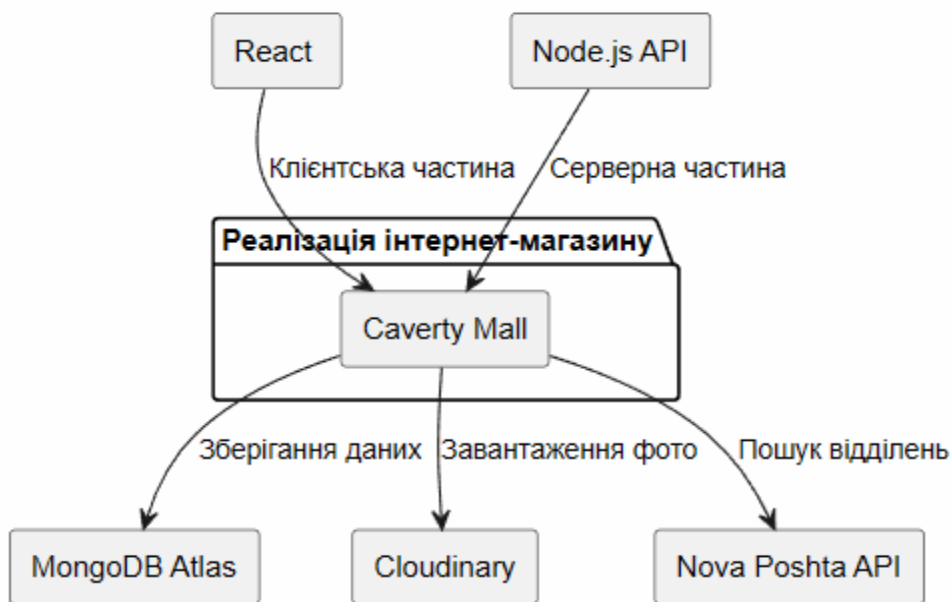


Рис. 2.8. Діаграму компонентів інтернет-магазину Caverty Mall

Головним компонентом системи є Caverty Mall, який поєднує клієнтську та серверну частини. Клієнтською частиною застосунку є React [2], до якого входять всі сторінки та компоненти. За серверну частину застосунку відповідає Node.js API [10], яка реалізує REST API [13], авторизацію [22] та обробку даних. Для збереження даних про товари, користувачів, замовлення, відгуки та побажання клієнтів використовується база даних MongoDB Atlas. Хмарний сервіс Cloudinary використовується для збереження зображень товарів, до яких

система має доступ через URL посилання. Також використаний зовнішній сервіс Nova Poshta API, для пошуку міста та відділення при оформленні замовлення користувачем [23].

Висновки до другого розділу

Під час розробки інтернет-магазину Caverty Mall було проаналізовано основні варіанти використання системи відповідно до поставлених вимог. На основі цього було сформовано архітектуру системи та визначено взаємодію між основними компонентами.

У процесі проєктування було розроблено структуру бази даних та взаємодію між її колекціями [18]. Було розроблено п'ять основних колекцій MongoDB, а саме: users; products; orders; reviews; suggestions. Для кожної колекції було сформовано структуру та призначення. Також було описано алгоритм роботи основних функцій системи, таких як авторизація користувача, фільтрація товарів, додавання товару адміністратором та оформлення замовлень.

Було описано реалізацію програми, наведено фрагмент коду авторизації, фільтрації товарів, завантаження зображень через Cloudinary, система відгуків з оновленням рейтингу, інтеграція з API Нової Пошти та кошика. Також було побудовано діаграму компонентів для відображення зв'язку між модулями системи та інтеграції з зовнішніми сервісами.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмотря О. Г.				45
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ-МАГАЗИНОМ ОДЯГУ CAVERTY MALL

3.1. Порядок встановлення та налаштування параметрів системи

Для працездатності інтернет-магазину одягу необхідно виконати розгортання всіх компонентів системи та налаштувати їхню спільну роботу. Важливо забезпечити роботу як кожного елемента окремо так і всю систему загалом. Для відображення архітектури розгортання було побудовано діаграму розгортання (рис. 3.1).

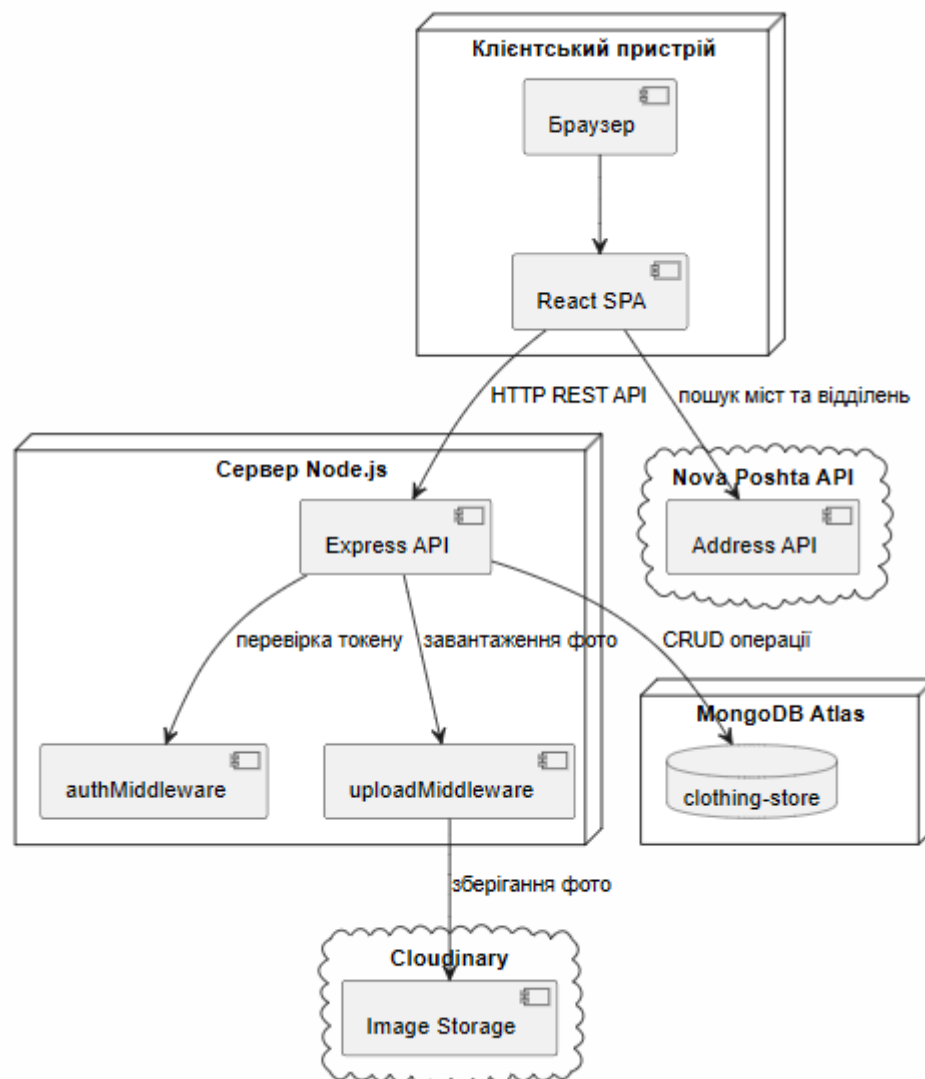


Рис. 3.1. Діаграма розгортання інтернет-магазину Saverty Mall

Розглянемо основні процеси розгортання системи. За клієнтську частину застосунку відповідає React SPA [11], який відображає інтерфейс у користувача та реалізовує зв'язок з серверною частиною через REST API [13]. Включає в себе всі сторінки магазину, компоненти, кошик та форми. Локальний запуск здійснюється за допомогою команди "npm run dev" [25].

За серверну частину застосунку відповідає Express API [32], який обробляє HTTP запити від клієнта, відповідає за реалізацію бізнес-логіки та взаємодію з базою даних. Запуск здійснюється за допомогою команди "npm run dev", запускається Node.js сервер [1], який автоматично перезапускається під час внесення змін.

Для того щоб користувачі мали різні рівні доступу було реалізовано два middleware authMiddleware та uploadMiddleware. authMiddleware містить дві функції, одна з них protect яка відповідає за перевірку наявності та чи є дійсним JWT токен при кожному запиті до захищеного функціоналу. Друга функція adminOnly перевіряє чи має користувач роль адміністратора. Якщо під час надсилання запиту токен буде відсутнім або недійсним, то система поверне помилку 401, якщо ж у користувача немає ролі адміністратора тоді буде помилка 403, клієнтська частина опрацює ці помилки та повідомить зрозумілим текстом. Саме тому звичайний користувач не зможе використати функціонал призначений для адміністратора.

uploadMiddleware відповідає за обробку завантажених файлів [29] та передає їх до сервісу Cloudinary для зберігання. Після того як зображення успішно завантажиться Cloudinary оптимізує його та поверне URL посилання, яке буде зберігатися в базі даних MongoDB. Завдяки чому завантаження зображення на сайті відбувається швидше.

MongoDB Atlas база даних в якій зберігаються усі дані системи інформація про товари, користувачів, відгуки та побажання покупців [17]. Взаємодія з базою даних відбувається завдяки бібліотеці Mongoose, яка перевіряє дані перед збереженням та робить неможливим збереження некоректних даних.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмотря О.Г.				47
Змн.	Арк.	№ докум.	Підпис	Дата		

Cloudinary - хмарний сервіс який використовується для збереження та оптимізації зображень товарів. Отримує зображення від сервера, обробляє їх та повертає URL посилання для збереження в базі даних.

Nova Poshta API - зовнішній сервіс який використовується на стороні клієнтської частини додатку для пошуку міста та відділення Нової Пошти при оформленні замовлення. Дані запити виконуються без використання серверної частини [23].

Для того щоб встановити систему спочатку необхідно створити обліковий запис в MongoDB Atlas для налаштування бази даних. Після чого необхідно створити обліковий запис в сервісі Cloudinary, щоб отримати необхідні ключі для подальшої взаємодії з сервісом. Необхідно отримати API ключ Нової Пошти для подальшої роботи з сервісом. Також є необхідним встановлення сучасного браузера.

Для запуску системи потрібно встановити Node.js версії 18 і вище та пакетний менеджер npm [1]. Потім необхідно клонувати даний репозиторій проєкту. Встановити усі необхідні залежності в папках "backend" та "frontend" командою "npm install". Після чого необхідно створити файл ".env" в папці "backend" та заповнити необхідні змінні.

```
PORT=5000
MONGODB_URI=посилання_на_MongoDB_Atlas
JWT_SECRET=секретний_ключ
CLOUDINARY_CLOUD_NAME=назва_хмари
CLOUDINARY_API_KEY=ключ_API
CLOUDINARY_API_SECRET=секрет_API
```

Кожна з цих змінних має свою роль. За порт на якому запускається серверна частина додатку відповідає змінна "PORT". За підключення до бази даних відповідає "MONGODB_URL". JWT_SECRET секретний та унікальний токен за допомогою якого перевіряється авторизація користувача. Змінні "CLOUDINARY_CLOUD_NAME", "CLOUDINARY_API_KEY" та "CLOUDINARY_API_SECRET" відповідають за підключення до хмарного сервісу для збереження зображень.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				48
Змн.	Арк.	№ докум.	Підпис	Дата		

Після чого необхідно запуснути серверну та клієнтську частину додатку командою "npm run dev". Після виконання всіх дій система буде готова до роботи. Серверна частина матиме порт 5000, а клієнтська 5173.

3.2. Структура інтерфейсу інтернет-магазину одягу Caverty Mall

При відкритті сайту користувач потрапляє на головну сторінку (рис. 3.2) де спершу бачить слайдер з актуальними колекціями та рекламними банерами, блок з рекомендованими товарами та блок з інформацією про магазин. Головна панель містить логотип магазину, пошук товарів, іконку кошика [30] та посилання на основні розділи, такі як: "Магазин"; "Про нас"; "Контакти"; "Список бажань".

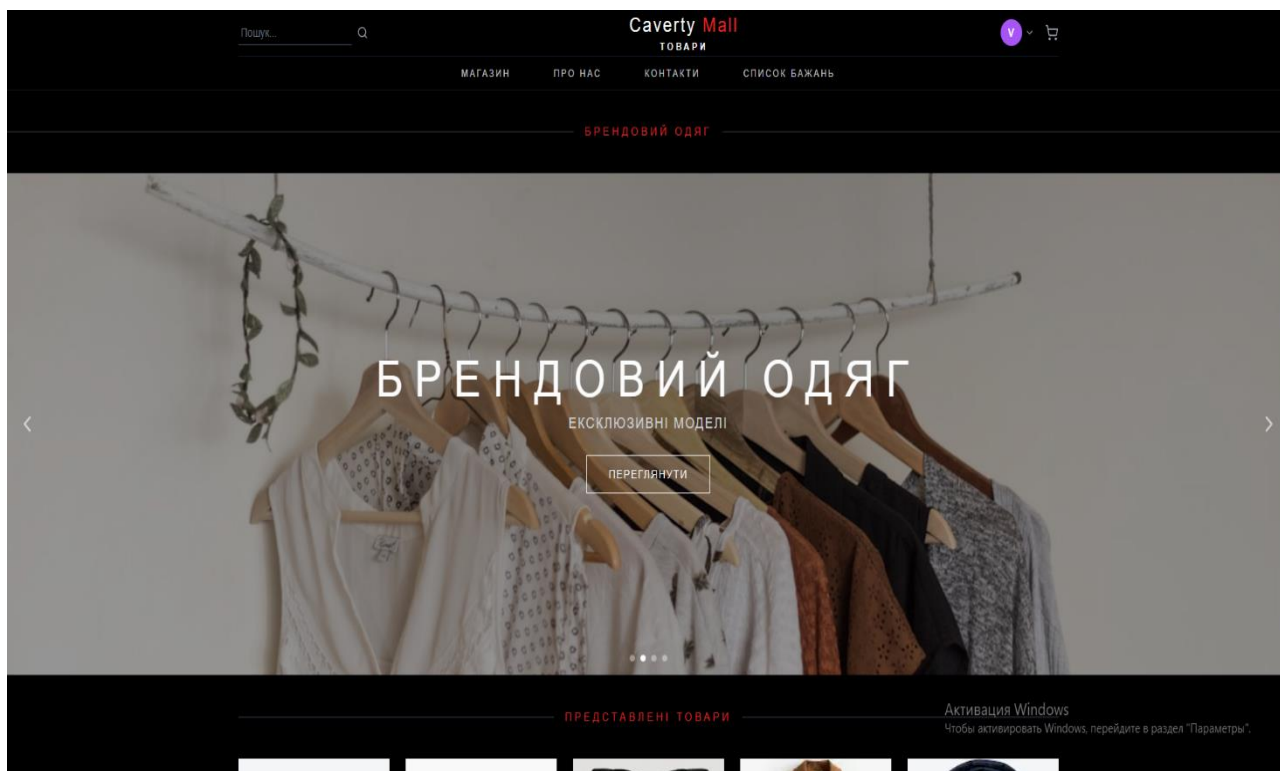


Рис. 3.2. Головна сторінка

На головній панелі також є випадаюче меню користувача. За допомогою нього покупець може зайти у власний профіль, переглянути власні замовлення. А для адміністратора в даному меню реалізована вкладка "Адмін панель", через яку здійснюється управління інтернет-магазином (рис. 3.3).

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

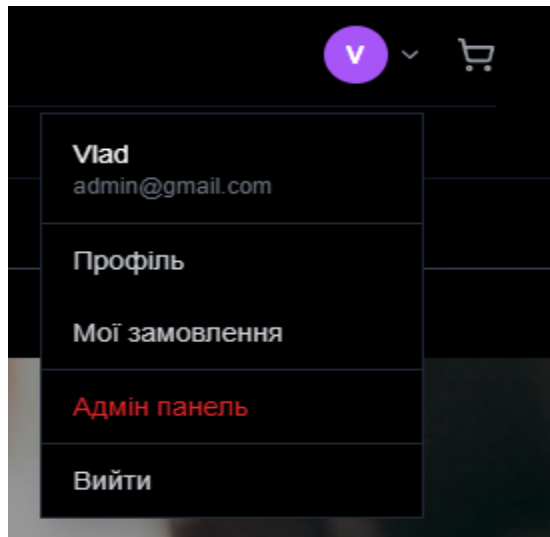


Рис. 3.3. Випадаюче меню користувача

З головного меню користувач може перейти у розділ "Магазин" де відображаються усі товари (рис. 3.4). На якому є можливість здійснення фільтрації за великою кількістю різноманітних характеристик. Також є сортування будь-яким зручним способом за зростанням та спаданням ціни, популярністю, новизною, рейтингом та по алфавіту. Також з даної сторінки можна одразу додати товар у кошик [24].

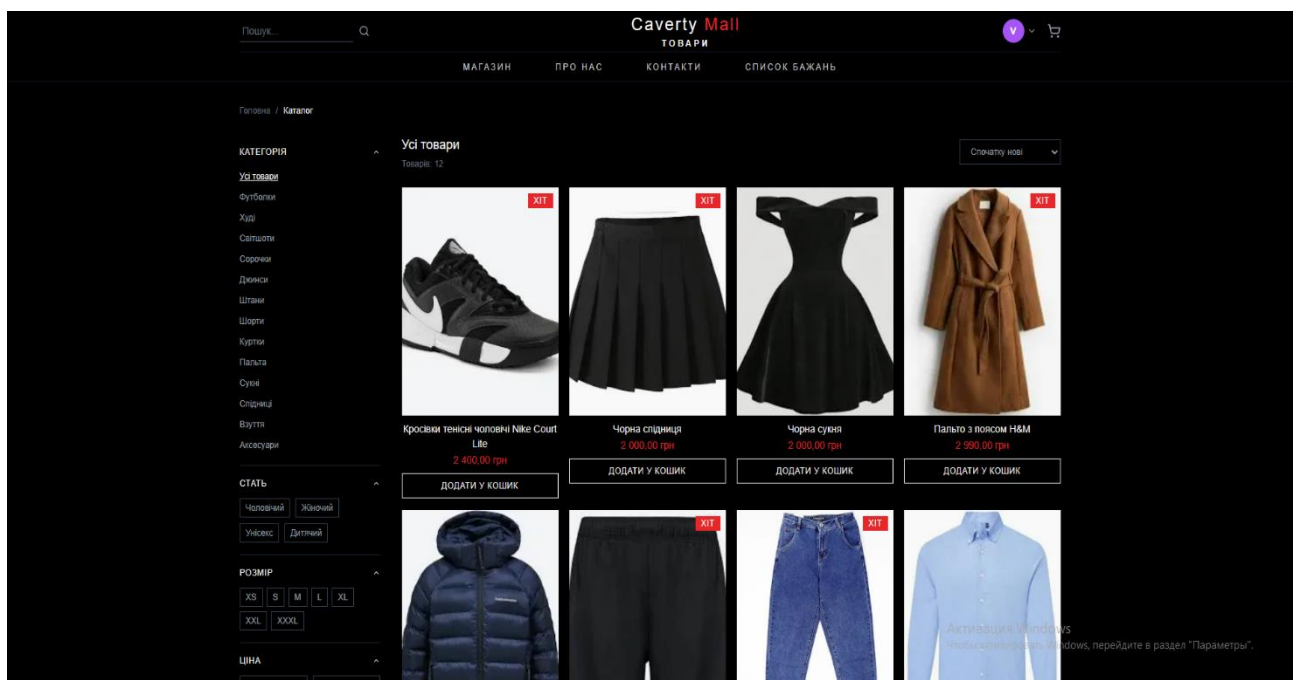


Рис. 3.4. Сторінка з відображенням товарів

Зі сторінки "Магазин" можна перейти на сторінку будь-якого товару, де зможемо побачити детальнішу та розширену інформацію про одяг, прочитати опис, обрати потрібний розмір, кількість, додати до кошика або купити. Якщо

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижевотря О.Г.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

якогось товару залишилося мало на складі користувача про це повідомляється, коли він обирає такий товар (рис. 3.5).

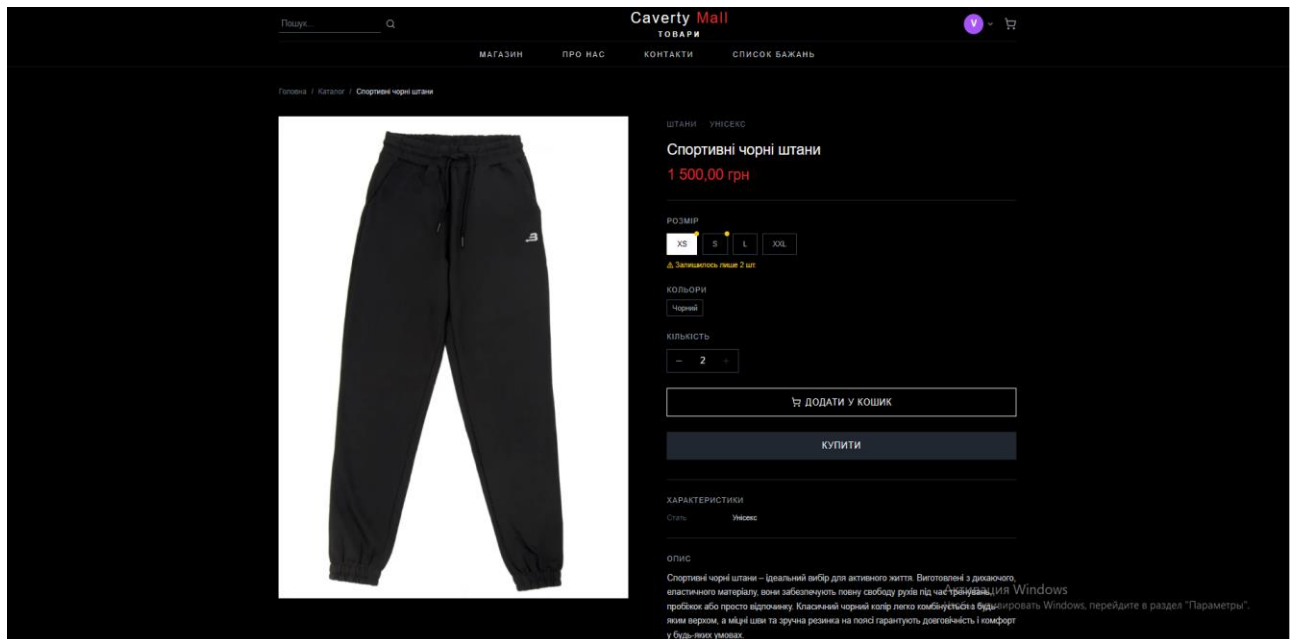


Рис. 3.5. Сторінка товару

Якщо натиснути кнопку "Додати товар у кошик" товар на сторінці якого знаходився буде додано до кошика. Після чого користувач може продовжити пошук збільшити кількість або перейти до оформлення замовлення (рис. 3.6).

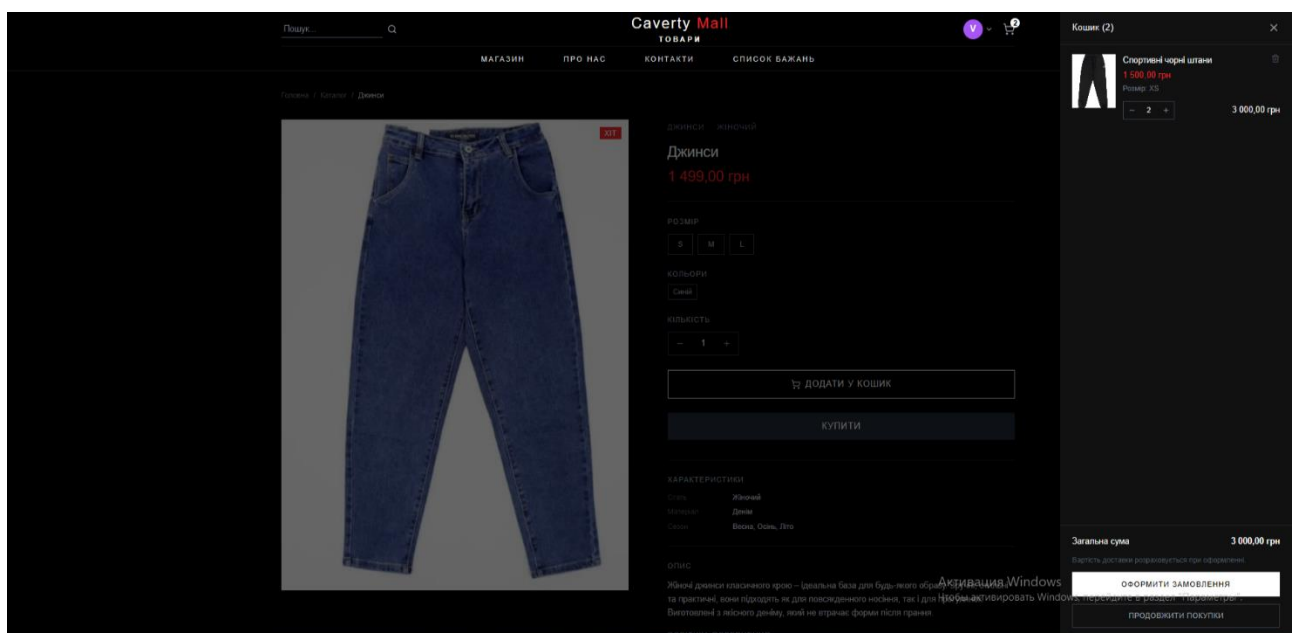


Рис. 3.6. Кошик з обраними товарами

Після того як покупець обрав усі товари які йому необхідно. Він може перейти до оформлення замовлення. Після чого необхідно ввести персональні дані, місто та відділення (рис 3.7)..

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				51
Змн.	Арк.	№ докум.	Підпис	Дата		

Пошук

Caverty Mail
ТОВАРИ

МАГАЗИН ПРО НАС КОНТАКТИ СПИСОК БАЖАНЬ

Головна / Оформлення замовлення

Оформлення замовлення

ОСОБИСТІ ДАНІ

ІМ'Я * Іван

ПРИЗВИЩЕ * Іванов

ТЕЛЕФОН * +380 683545326

СПОСІБ ДОСТАВКИ

☒ Нова Пошта

☐ Укрпошта

МІСТО * м. Житомир, Житомирська обл.

ВІДДІЛЕННЯ НП * Відділення НП: вул. Гранітна, 16

СПОСІБ ОПЛАТИ

☒ Наглядний платіж

☐ Оплата картою (наразі недоступно)

ВАШЕ ЗАМОВЛЕННЯ

Сторінки, черга оплати 3 000 ₴

Розмер: XS

Висота: 3

Товари 3 000 ₴

Доставка за тарифом порівняння

Разом 3 000 ₴

ОФОРМИТИ ЗАМОВЛЕННЯ

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Рис. 3.7. Сторінка оформлення замовлення

Після успішного оформлення замовлення користувача буде повідомлено про це (рис. 3.8). Замовлення відразу буде збережено та відображено в адміністративній панелі.

Пошук

Caverty Mail
ТОВАРИ

МАГАЗИН ПРО НАС КОНТАКТИ СПИСОК БАЖАНЬ

✓

Замовлення оформлено!

Дякуємо за покупку. Ми зв'язалися з вами найближчим часом.

МОЇ ЗАМОВЛЕННЯ

ПРОДОВЖИТИ ПОКУПКУ

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

Рис. 3.8. Повідомлення про успішне оформлення замовлення

Але перед тим як оформити замовлення система перевіряє чи користувач авторизований у системі, тому що здійснювати покупки можливо лише за цієї умові. Якщо у покупця вже є зареєстрований акаунт, він може просто в нього авторизуватися (рис. 3.9).

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

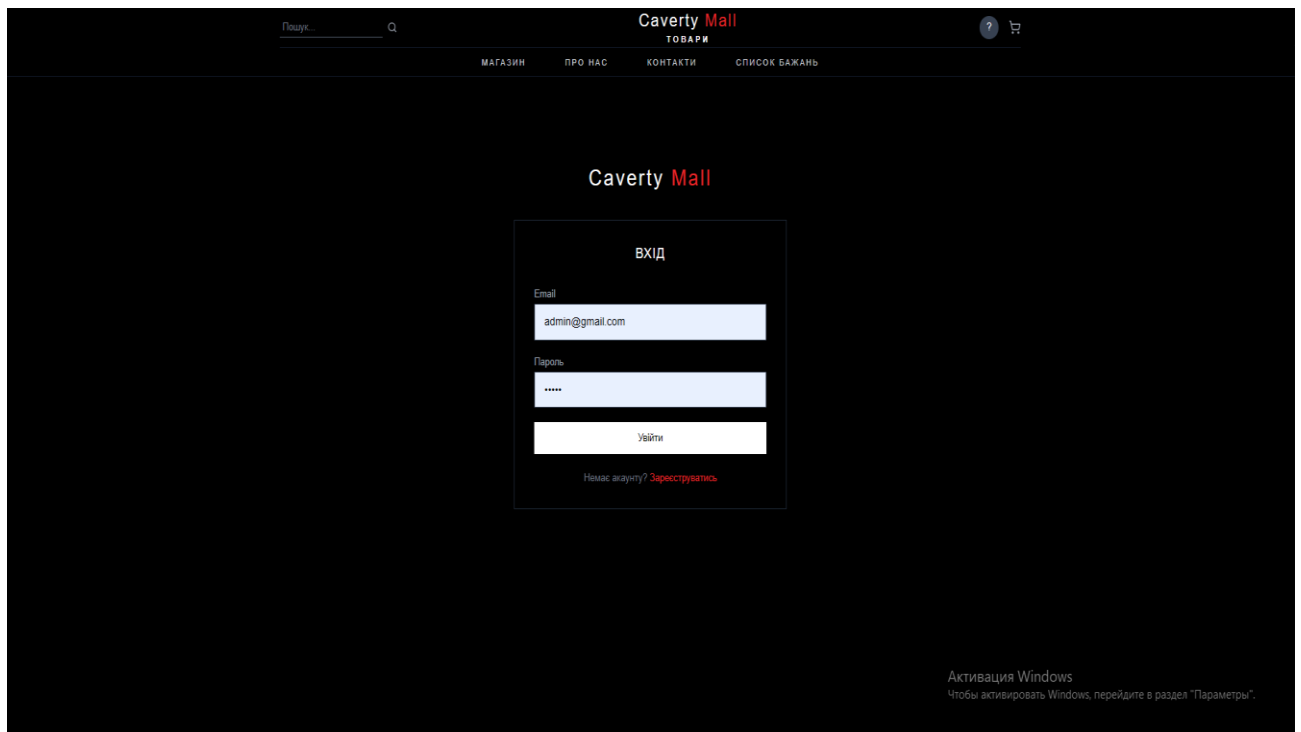


Рис. 3.9. Сторінка для авторизації користувачів

Якщо ж у користувача немає створеного акаунту, він може зареєструвати новий (рис. 3.10), де йому необхідно буде ввести ім'я, пошту та пароль [28]. В формі також присутня валідація на перевірку введених даних.

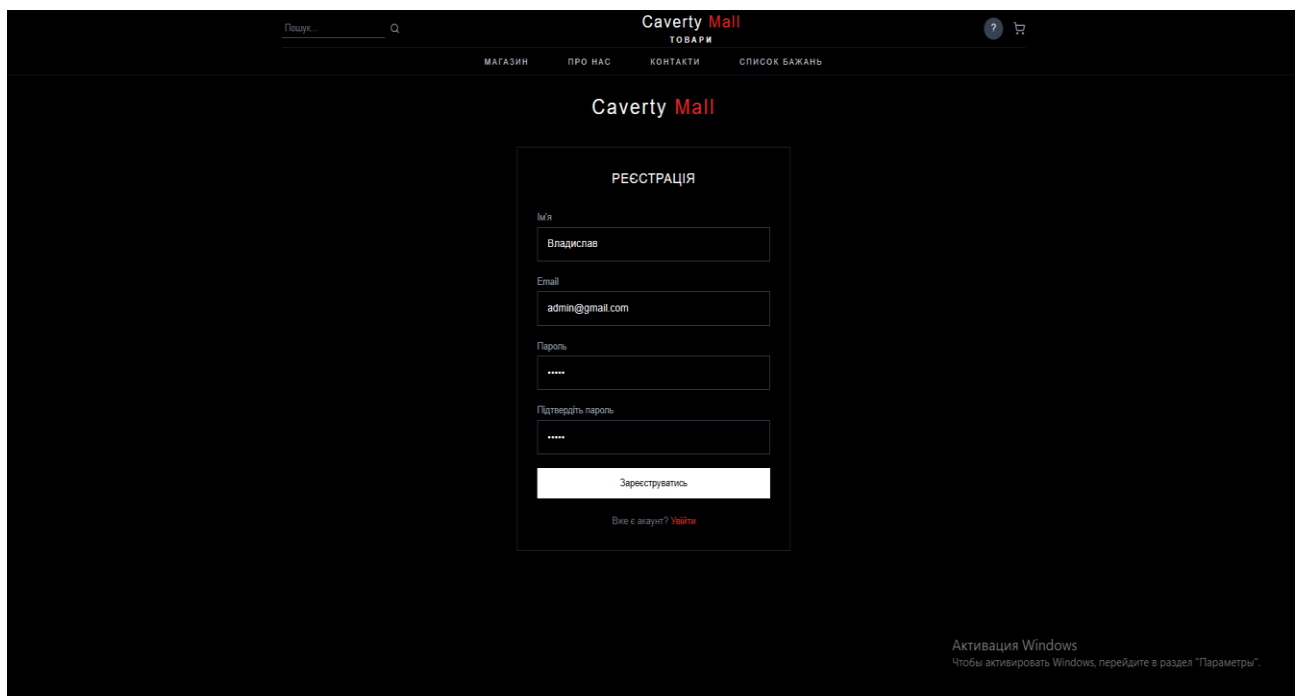


Рис. 3.10. Сторінка для реєстрації користувачів

Також є сторінка "Про нас" на якій розповідається історія магазину, його цінності та переваги (рис. 3.11).

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О. Г.				53
Змн.	Арк.	№ докум.	Підпис	Дата		

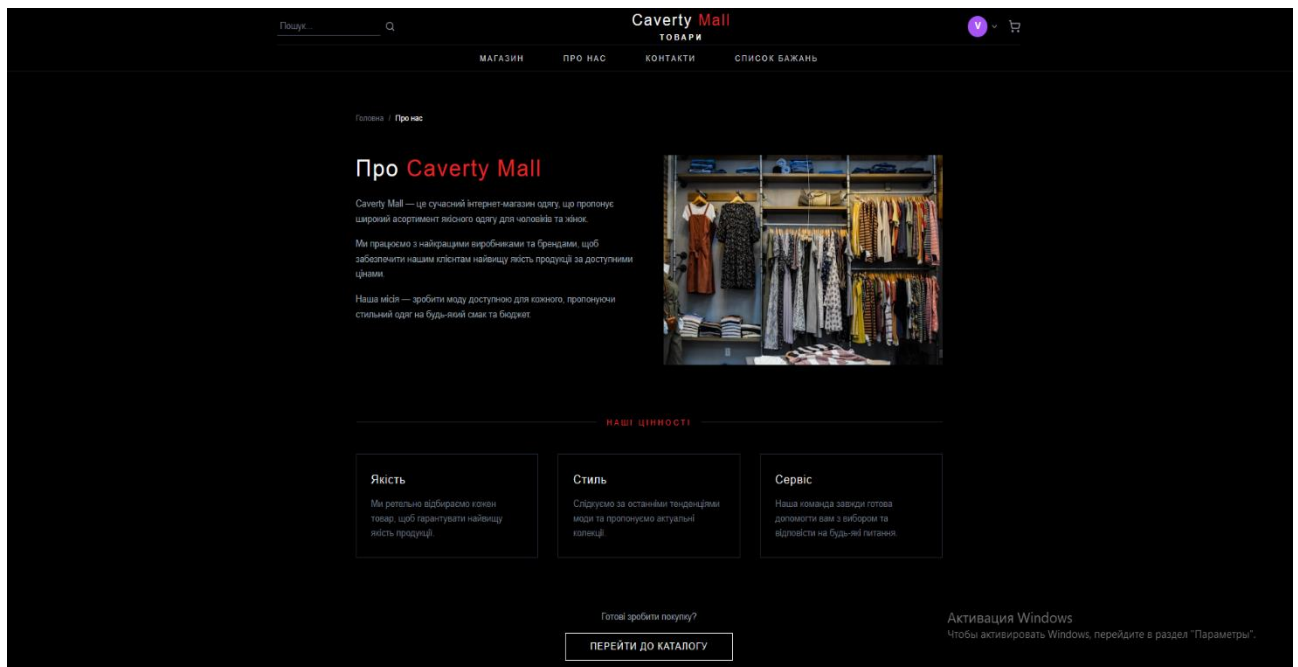


Рис. 3.11. Особиста сторінка інтернет-магазину "Про нас"

Також є сторінка "Контакти" на якій відображені методи зв'язку з адміністрацією магазину або можна написати повідомлення напряму з платформи, заповнивши відповідну форму (рис. 3.12).

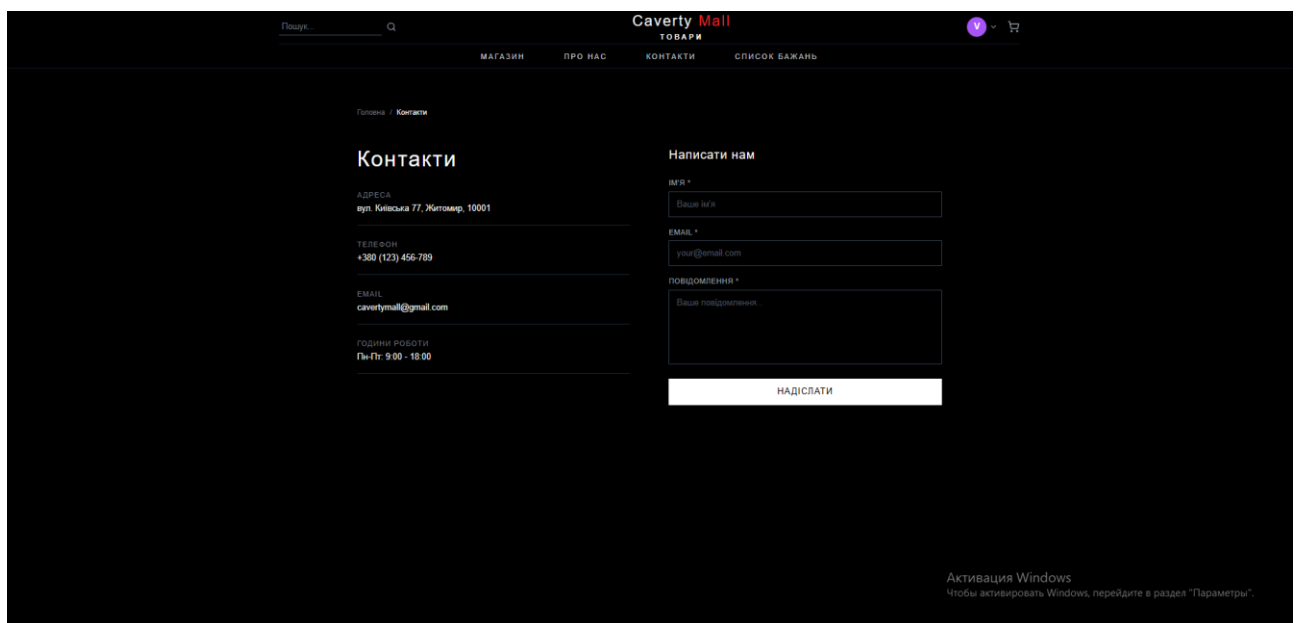


Рис. 3.12. Сторінка "Контакти"

На платформі присутня можливість користувачам пропонувати товари, які б вони хотіли бачити в майбутньому в магазині. Для цього їм потрібно перейти на сторінку "Список бажань" заповнити необхідну форму, де необхідно вказати назву одягу, категорію, бажаний ціновий діапазон, додати опис за бажанням та

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижевотря О.Г.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

додати фото для прикладу. Після чого побажання надійде адміністрації сайту, яка зможе надати статус побажання та це відразу відобразиться (рис. 3.13).

Рис. 3.13. Сторінка "Список бажань"

Для управління інтернет-магазином була розроблена адмін-панель, яка доступна лише для користувачів з роллю адміністратор, функціоналом якої є відображення статистики, управління товарами, користувачами, список усіх замовлень та побажання (рис. 3.14).

Рис. 3.14. Адмін-панель для управління інтернет-магазином

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмотря О.Г.				55
Змн.	Арк.	№ докум.	Підпис	Дата		

У вкладці "Товари" відображаються усі додані товари з їх коротким описом є можливість їх редагувати та видаляти (рис. 3.15).

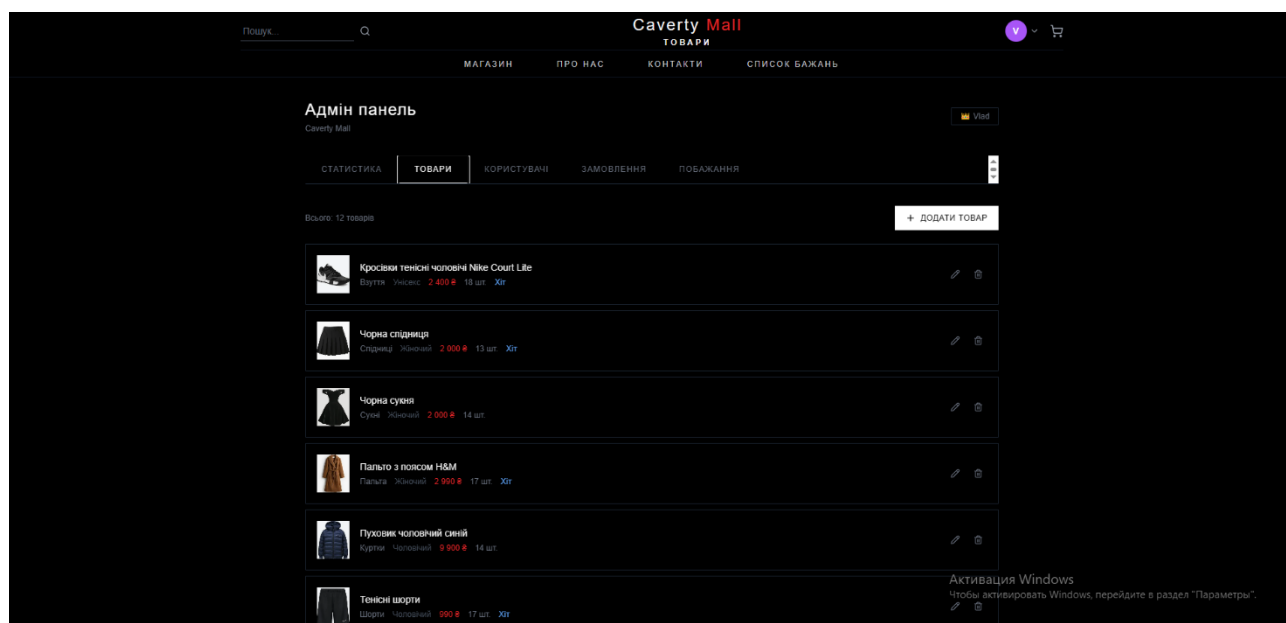


Рис. 3.15. Адмін-панель вкладки "Товари"

В даній вкладці також є можливість додавання товарів в магазин, для цього необхідно вказати назву, опис, ціну, бренд, категорію, стать, матеріал, сезон для якого призначений, розміри які в наявності, колір та завантажити зображення (рис. 3.16).

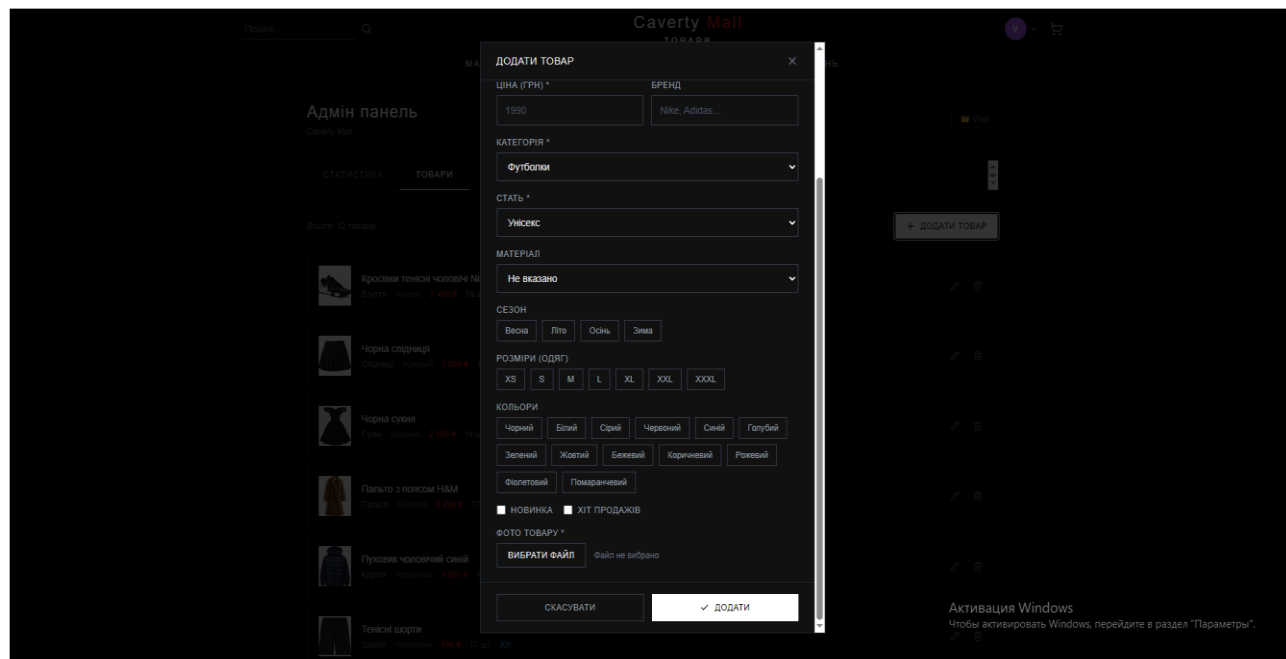


Рис. 3.16. Адмін-панель вкладки "Товари" додавання товарів

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

У вкладці "Користувачі" відображаються усі хто зареєструвався на сайті, в даній вкладці також є можливість надати роль адміністратора або забрати його (рис. 3.17).

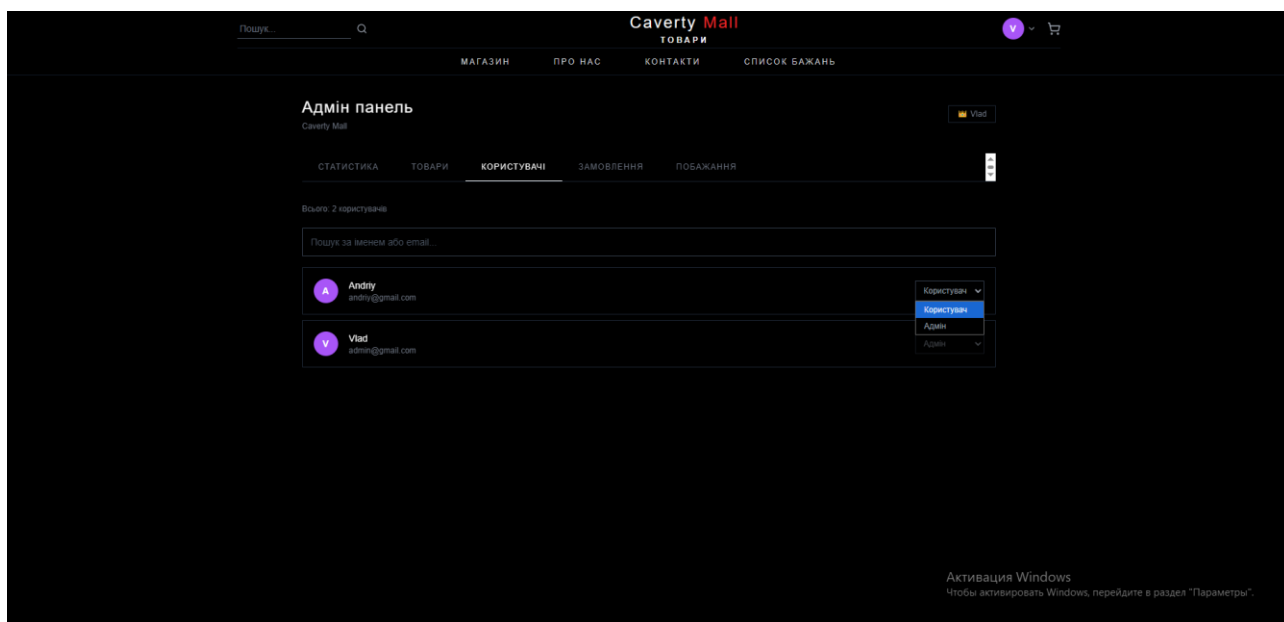


Рис. 3.17. Адмін-панель вкладки "Користувачі"

Для опрацювання замовлень було реалізовано вкладку "Замовлення" в якій відображаються усі замовлення та їхній статус. Всього існує 6 станів замовлення очікує, в обробці, відправлено, доставлено, скасовано та завершено. Якщо покупець відмінив замовлення до отримання тоді на товарі буде додаткова позначка, для додаткового привернення уваги (рис. 3.18).

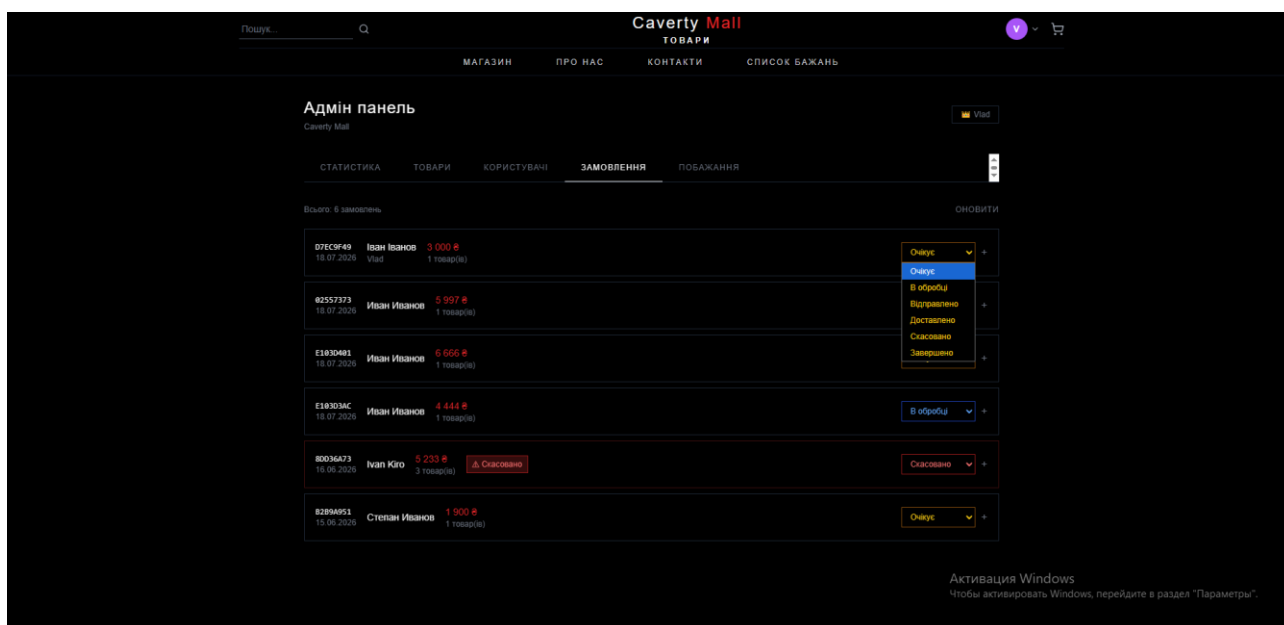


Рис. 3.18. Адмін-панель вкладки "Замовлення"

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

На сторінці з побажаннями відображено усі побажання, які надіслали користувачі. Адміністратор може змінити статус такого побажання, щоб користувач який це запропонував очікував даний товар або може одразу його додати (рис. 3.19).

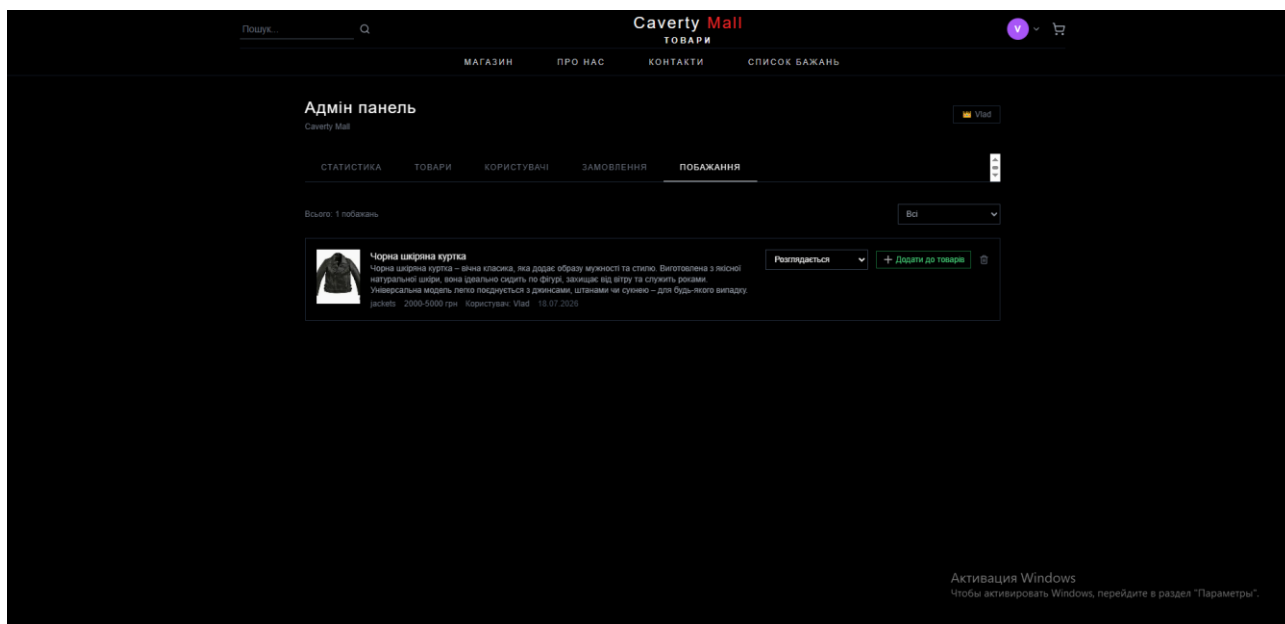


Рис. 3.19. Адмін-панель вкладка "Побажання"

Також на сторінці з кожним товаром є можливість залишати відгук з рейтингом його можна редагувати та видалити, є можливість відповідати на чужі коментарі (рис. 3.20). Адміністратор може видалити чужий відгук за порушення правил магазину.

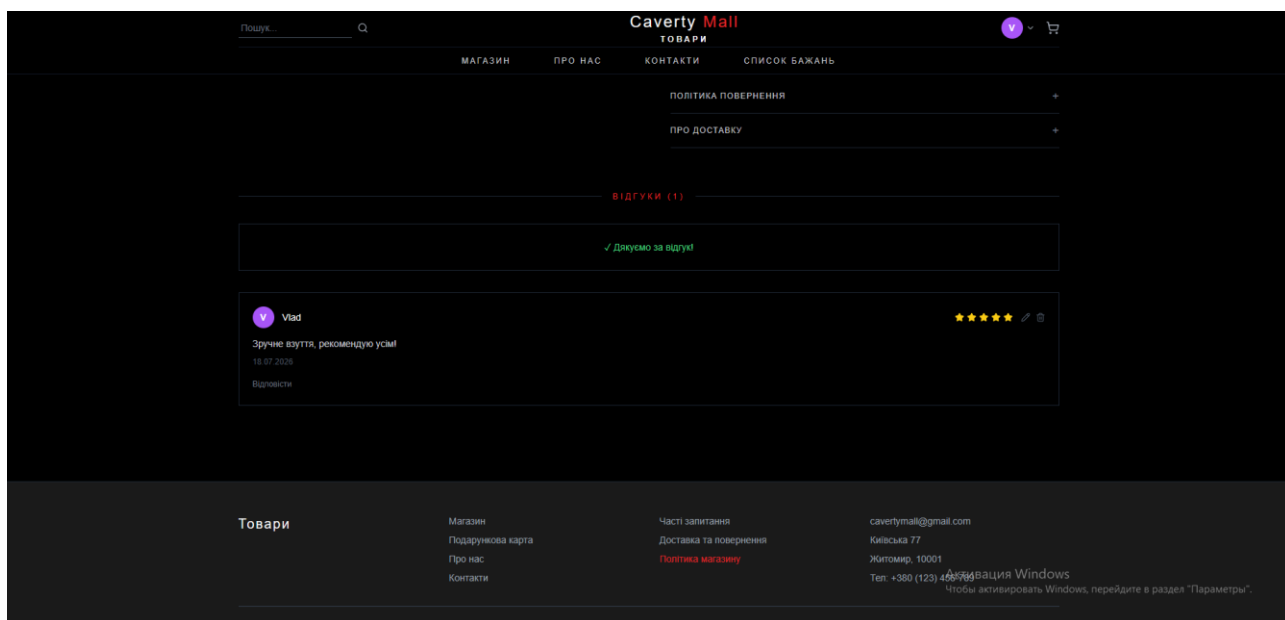


Рис. 3.20. Відгуки до товарів

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижевотря О.Г.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

3.3. Тестування роботи програмного комплексу

Тестування програмного продукту є важливим етапом розробки, це дозволяє виявити та усунути помилки в системі перед введенням її в експлуатацію. В ході тестування інтернет-магазину одягу Caverty Mall виконувалася перевірка на відповідність функціоналу визначеним вимогам.

Першим етапом тестування є перевірка реєстрації та авторизації користувачів. Для тестування було введено ім'я користувача, електронну пошту та пароль та натиснути кнопку "Зареєструватися". Після чого система успішно створила новий обліковий запис та перенаправила на головну сторінку. Якщо спробувати зареєструватися на електронну пошту яка уже зареєстрована в системі, відобразиться повідомлення про помилку (рис. 3.21).

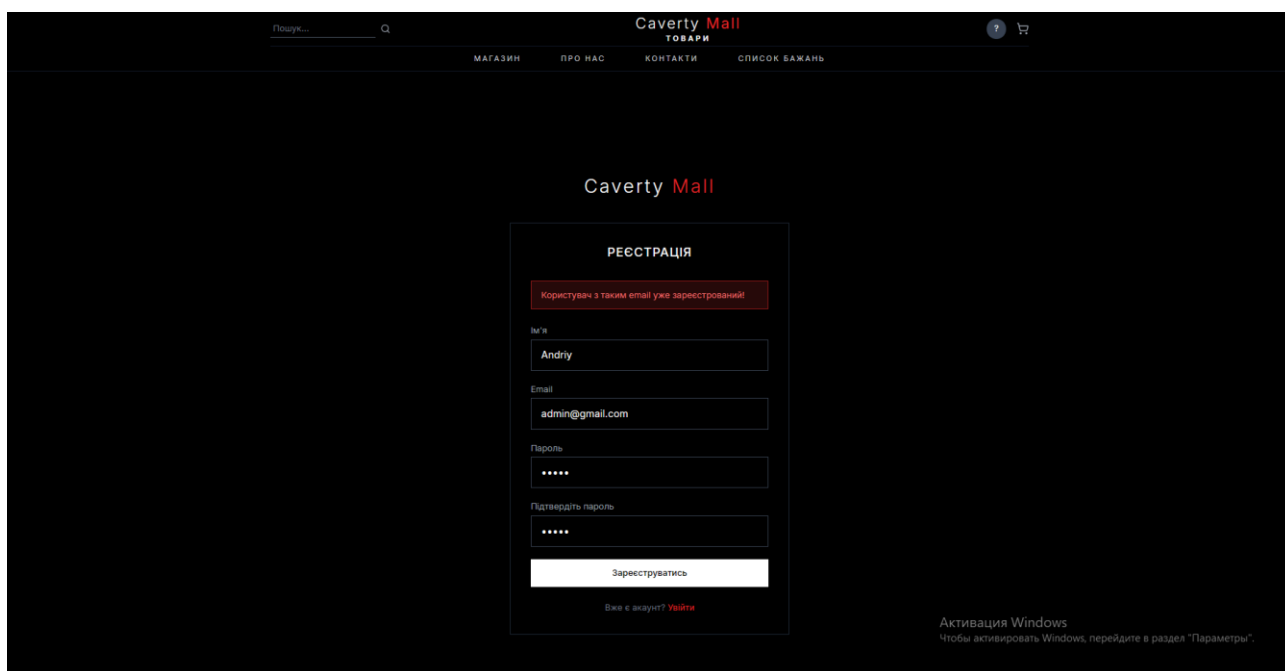


Рис. 3.21. Помилка при введенні email який уже використовується

Перевірка також відбувається на правильність введення паролю мінімальна кількість символів 5 [28]. Якщо паролі не збігаються повідомляється про помилку.

Якщо при авторизації користувач вводить правильні дані, система успішно виконує вхід та відображає всю підв'язану інформацію до нього [22]. У разі якщо пароль або email не вірні буде повідомлено про помилку (рис. 3.22).

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О. Г.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

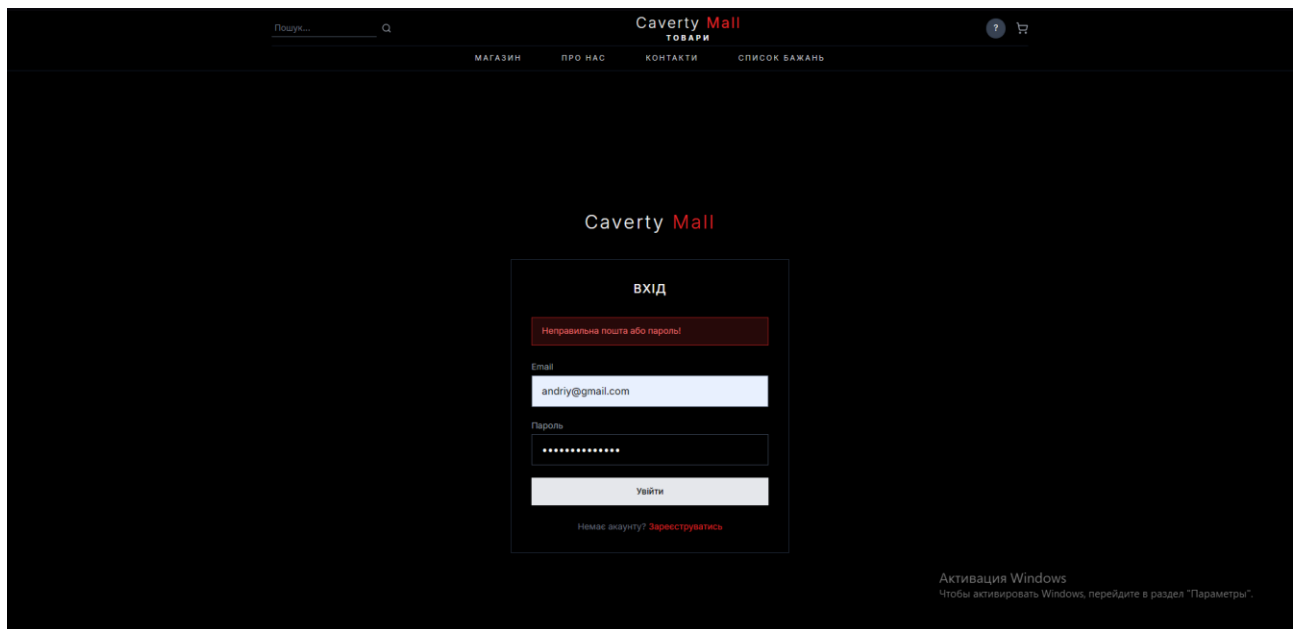


Рис. 3.22. Помилка авторизації при введенні не вірного email чи паролю

Важливим етапом є тестування сторінки каталогу та фільтрації. Було перевірено систему фільтрації за декількома параметрами, щоб одяг був для чоловіка та щоб розмір "M" був в наявності. Також було застосовано сортування "Спочатку дешевші" (рис. 3.23).

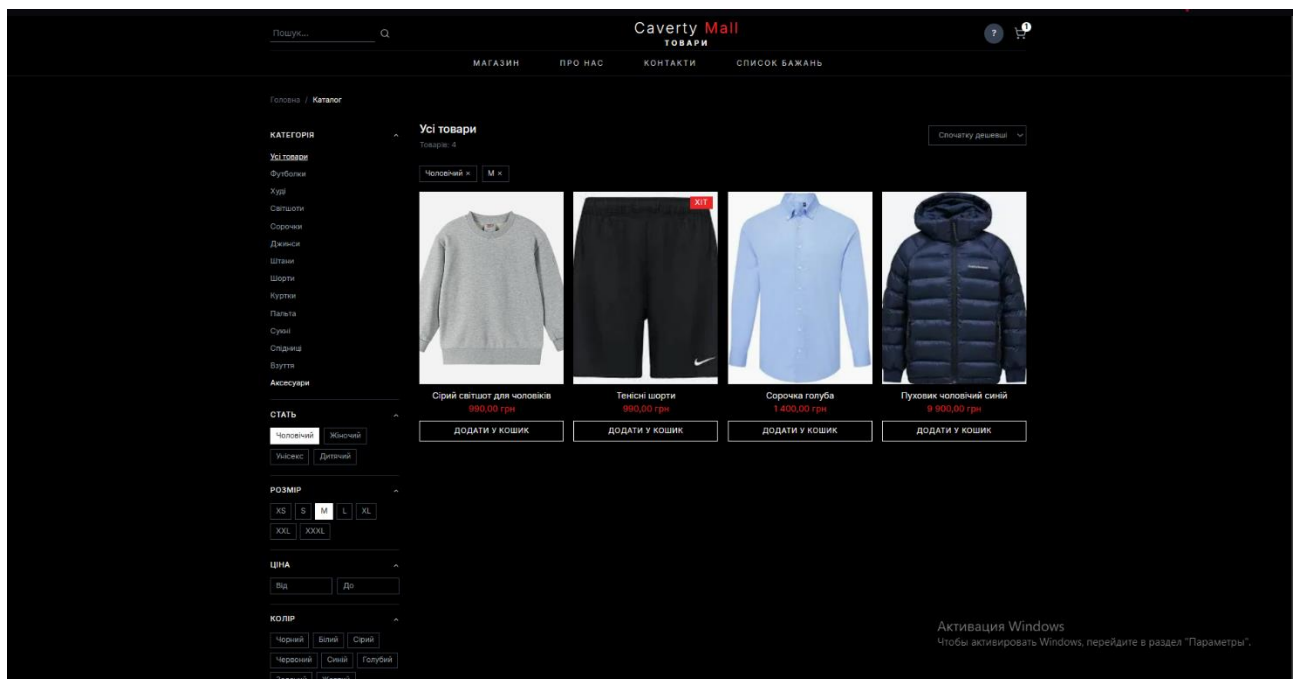


Рис. 3.23. Застосування фільтрів та сортування

Також був реалізований залишок на складі, щоб покупець не міг додати більшу кількість одягу чим є в наявності. Для кожного розміру є своя кількість

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмотря О.Г.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

залишку (рис. 3.24). Після здійснення кожного оформлення замовлення кількість товарів оновлюється.

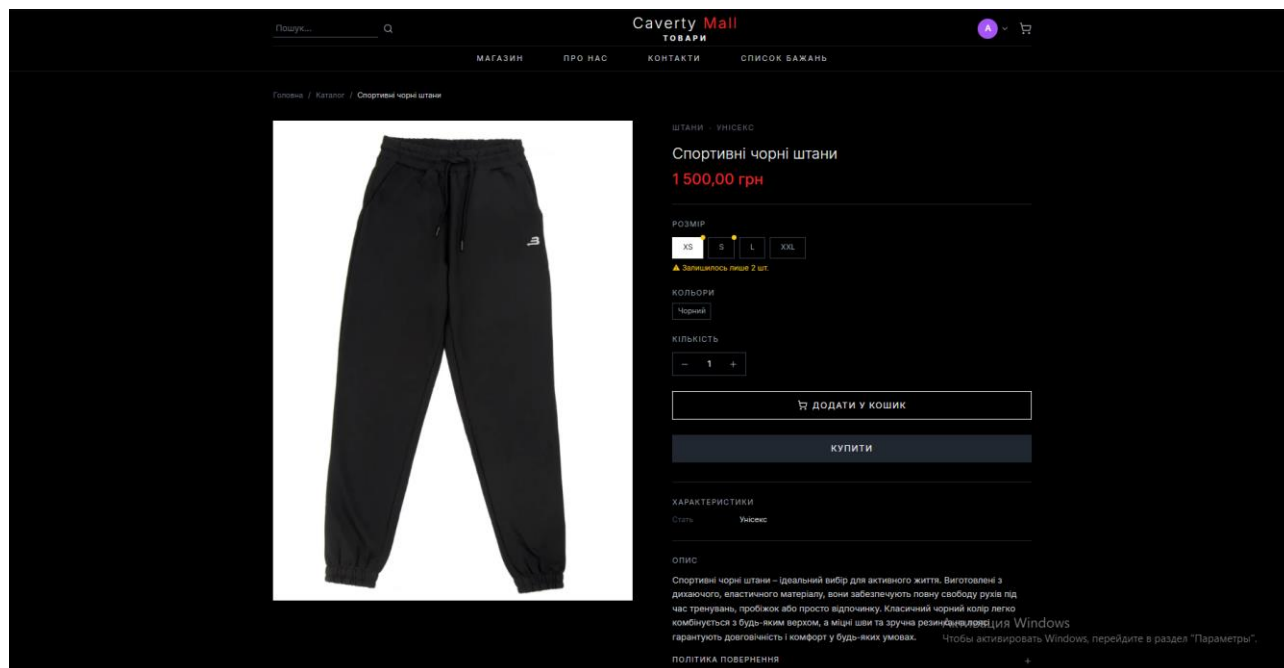


Рис. 3.24. Обмеження товарів яких залишилося мало

Коли неавторизований користувач обрав товари, які йому сподобалися і хоче здійснити замовлення система попросить його увійти в систему. Адже незареєстрований користувач не може оформити замовлення (рис. 3.25).

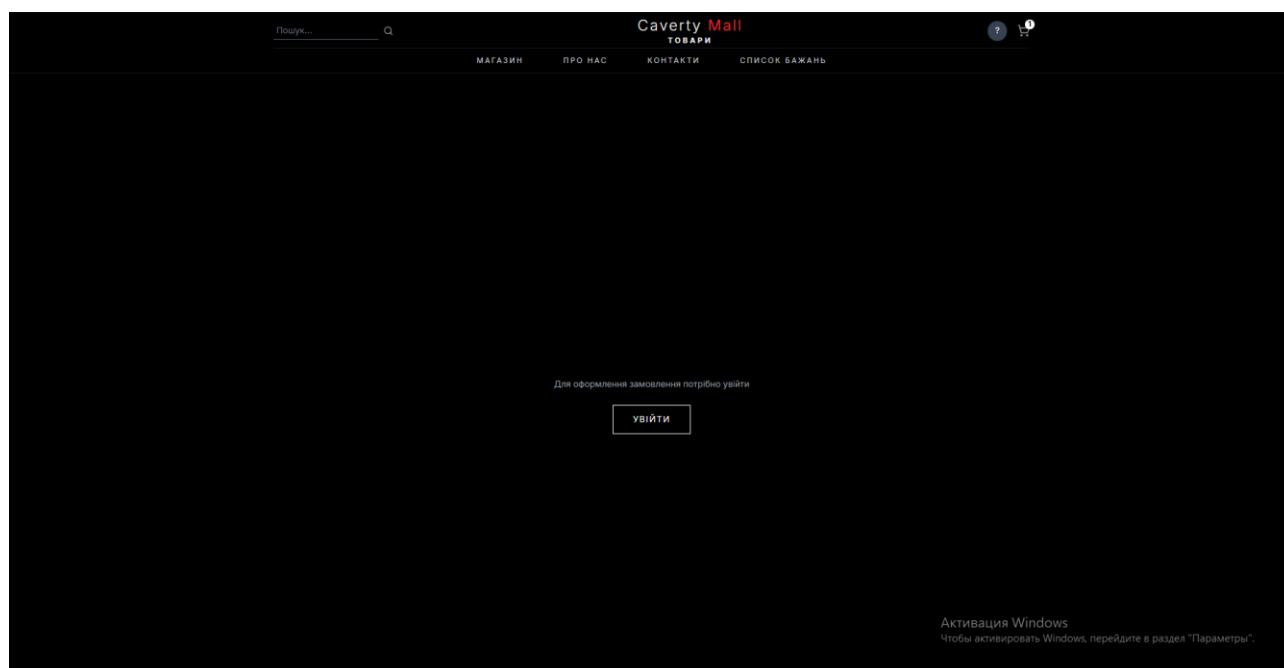


Рис. 3.25. Помилка при оформленні замовлення для неавторизованого користувача

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

При тестуванні особистого кабінету було перевірено відображення історії замовлень з можливістю переглянути деталі кожного замовлення та скасувати своє замовлення за потреби (рис. 3.26). Редагування профілю працює коректно, можна змінити ім'я та електронну пошту.

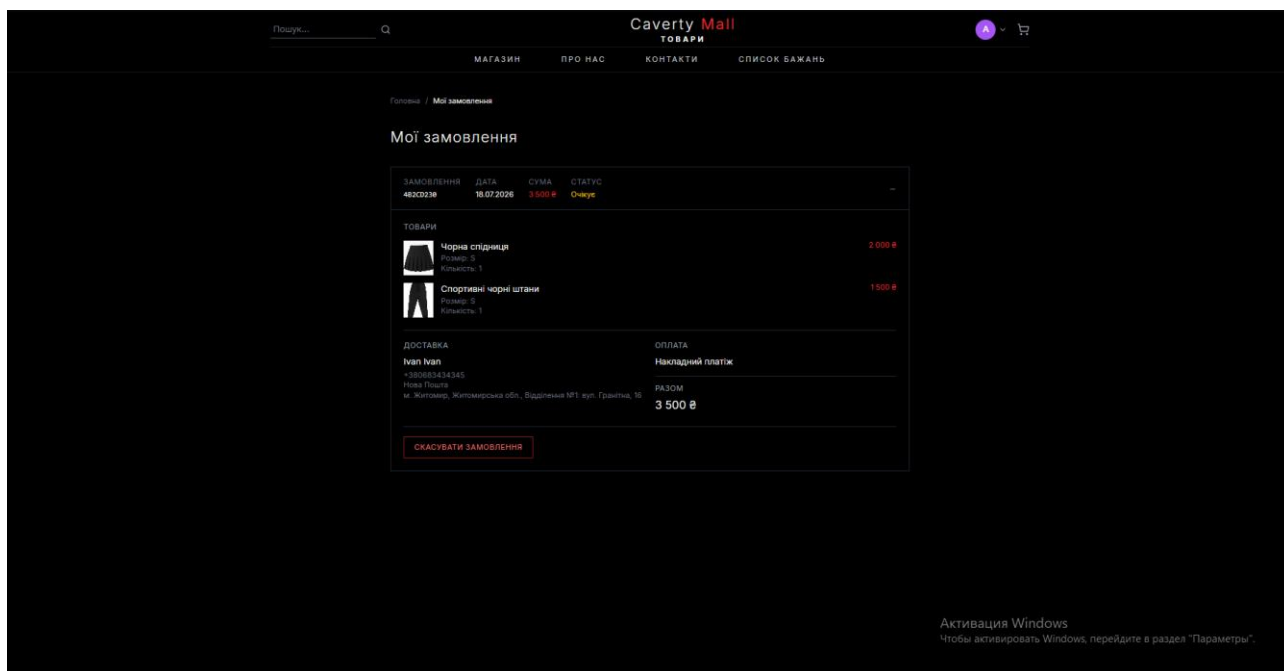


Рис. 3.26. Особистий кабінет відображення замовлень покупця

Також було протестовано адміністративну панель додано новий товар з завантаженням зображення, товар успішно додався у базу даних та відобразився у каталозі. Редагування існуючих товарів та їх видалення працює коректно (рис. 3.27). Зміна статусу замовлень та роль користувачів працюють.

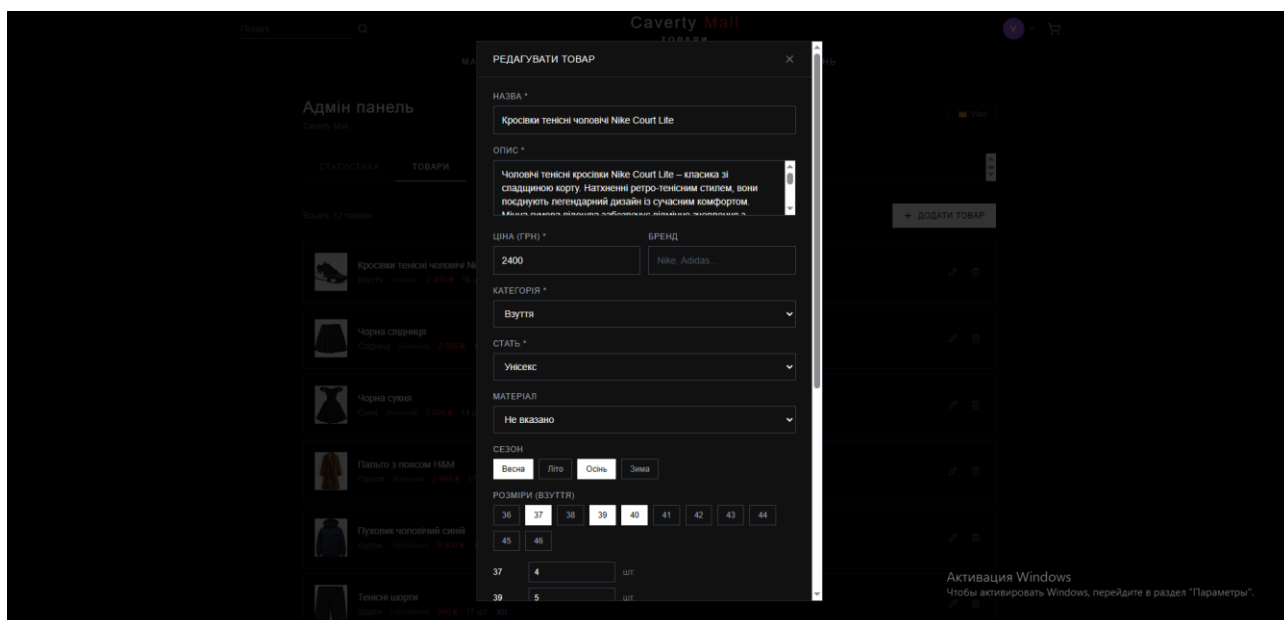


Рис. 3.27. Редагування уже існуючого товару

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмотря О.Г.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

Після завершення тестування було зображено таблицю з результатами тестування (табл. 3.1).

Таблиця 3.1

Результати тестування

Функція	Очікуваний результат	Фактичний результат	Статус
Реєстрація нового користувача	Створення нового акаунту та вхід в систему	Акаунт створено успішно	Пройдено
Реєстрація нового користувача з невірними даними	Повідомлення про помилку	Помилку відображено	Пройдено
Авторизація з правильними даними	Вхід в систему	Успішний вхід	Пройдено
Авторизація з невірними даними	Повідомлення про помилку	Помилка відображена	Пройдено
Фільтрація за категоріями	Відображення відібраних товарів	Відображено відфільтровані товари	Пройдено
Фільтрація за декількома критеріями(категорія, розмір, ціна та ХІТ товари)	Відображення відфільтрованих наявних товарів	Фільтрація не спрацювала для рейтингу, особливостей та матеріалу	Виправлено
Додавання товару до кошика та взаємодія з ними	Додавання товару до кошика та можливість змінювати кількість	Товари додано до кошика та можна змінювати кількість	Пройдено
Обмеження кількості товарів у кошику	Кількість товарів на складі обмежено	В кошику можна було додати необмежену кількість товарів	Виправлено
Оформлення замовлення (авторизований користувач)	Створення замовлення	Замовлення створюється	Пройдено
Оформлення замовлення (неавторизований користувач)	Повідомлення про помилку з проханням авторизуватися	Помилку відображено	Пройдено
Пошук міста та відділення через API Нової Пошти	Відображення міст та доступних відділень	Міста та відділення відображено	Пройдено
Оновлення залишку після замовлення	Залишок товару зменшується після замовлення	Залишок не оновлювався після замовлення	Виправлено
Залишення відгуків	Відгук зберігається та відображається на сторінці товару, рейтинг автоматично оновлюється	Відгук збережено, рейтинг оновлено, виявлено проблему створення пустого відгуку	Виправлено

Скасування замовлення	Скасування замовлення в системі та зміна статусу	Статус змінено та повідомлено адміністратора	Пройдено
Додавання товару адміністратором	Додавання товару в базу даних, та відображення у каталозі товарів	Товар додано та успішно відображається	Пройдено
Зміна статусу замовлення (адміністратором)	Зміна статусу замовлення у системі та в кабінеті покупця	Статус оновлено	Пройдено
Зміна ролі користувача (адміністратором)	Зміна ролі в системі та наданих функціонал користувачеві	Роль успішно змінено	Пройдено
Заповнення форми на сторінці "Список бажань"	Бажаний товар відображається у адміністратора	Товар відображено	Пройдено

Висновки до третього розділу

В третьому розділі було описано процес розгортання інтернет-магазину одягу Caverty Mall. Реалізовано діаграму розгортання на якій відображено взаємодію між клієнтською частиною додатку на React, серверною частиною на Node.js, базою даних MongoDB та зовнішніми сервісами Cloudinary і API Нової Пошти. Наведено порядок встановлення елементів системи з переліком необхідних команд та змінних.

Було описано структуру інтерфейсу користувача. Зображено основні сторінки застосунку, а саме головну сторінку, каталог товарів з фільтрацією та сортуванням, сторінку окремого товару з системою відгуків, кошик, форму оформлення замовлення, особистий кабінет покупця та адміністративну панель.

Було проведено функціональне тестування програмного продукту. Було перевірено роботу системи реєстрації, авторизації, фільтрації товарів, оформлення замовлення, систему відгуків та роботу адміністративної панелі. Після завершення тестування всіх систем можна дійти висновку що всі функції системи працюють правильно.

ВИСНОВКИ

В результаті виконання випускної кваліфікаційної роботи було розроблено функціональний інтернет-магазин одягу Caverty Mall, побудований на основі клієнт-серверної архітектури з використанням сучасних технологій.

В процесі виконання було проаналізовано аналоги у сфері онлайн-торгівлі одягом. Розглядалося три українських інтернет-магазини Remix, Goldi та Garda. Було виявлено їх основні недоліки: обмежений функціонал фільтрів товарів; відсутність повноцінного особистого кабінету покупця; система відгуків або відсутня або реалізована окремою сторінкою, що створює незручності. Тому вони були враховані при розробці системи.

Було обрано та обґрунтовано технології для розробки платформи. Клієнтська частина додатку реалізована з використанням бібліотеки React та TypeScript. Серверна частина додатку реалізована на Node.js та Express. Для збереження даних було використано базу даних MongoDB. Для збереження зображень було використано хмарний сервіс Cloudinary. Авторизацію було реалізовано за допомогою JWT токенів.

Спроектowana архітектура додатку та структура бази даних. База даних складається з п'яти колекцій: users; products; orders; reviews; suggestions. Також було побудовано діаграму варіантів використання, класів та алгоритмів роботи системи.

Серверну частину було реалізовано з REST API, яка відповідає за маршрути для керування товарами, замовленнями, користувачами, відгуками та побажаннями покупців.

Розроблена клієнтська частина додатку включає в себе: каталог з товарів з розширеною системою фільтрації за категоріями, статтю, розміром, кольором, матеріалом, сезоном, та ціновим діапазоном; сторінку товару з зображенням та відображенням залишку товарів, яких залишилося мало; кошик зі збереженням між сесіями користувача; форму для оформлення замовлення; особистий кабінет користувача; систему відгуків; розділ "Список бажань"; адміністративну панель.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

Реалізовано систему реєстрації та авторизації користувачів з різними правами доступу. Передбачено дві ролі покупець та адміністратор, для кожного з них доступний свій функціонал. Було проведено тестування системи, за результатами якого працювало все правильно.

Цінність розробленої системи полягає в можливості її використання малим та середнім бізнесом у сфері роздрібної торгівлі одягом в мережі інтернет. Система надає зручний інтерфейс для керування асортиментом товарів та замовленнями, а для покупців реалізовано зручний інтерфейс для здійснення пошуку та покупок товарів.

Подальшим вдосконаленням програмного комплексу може бути: інтеграція платіжної системи на сайті для оплати замовлень; реалізувати сповіщення покупців про зміну статусу замовлення на пошту; додати функціонал порівняння товарів; реалізувати систему знижок та промокодів.

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмотря О. Г.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Flanagan D. JavaScript: The Definitive Guide. 7th ed. Sebastopol : O'Reilly Media, 2020. 706 p.
2. Banks A., Porcello E. Learning React. 2nd ed. Sebastopol : O'Reilly Media, 2020. 310 p.
3. Bradshaw S., Brazil E., Chodorow K. MongoDB: The Definitive Guide. 3rd ed. Sebastopol : O'Reilly Media, 2019. 514 p.
4. Brown E. Web Development with Node and Express. 2nd ed. Sebastopol : O'Reilly Media, 2019. 346 p.
5. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. Sebastopol : O'Reilly Media, 2021. 612 p.
6. Richardson L., Ruby S. RESTful Web Services. Sebastopol : O'Reilly Media, 2007. 448 p.
7. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 560 p.
8. Freeman A. Pro React 16. New York : Apress, 2019. 750 p.
9. Cherny B. Programming TypeScript: Making Your JavaScript Applications Scale. Sebastopol : O'Reilly Media, 2019. 324 p.
10. Casciaro M., Mammino L. Node.js Design Patterns. 3rd ed. Birmingham : Packt Publishing, 2020. 664 p.
11. Mesbah A., Van Deursen A. Migrating Multi-page Web Applications to Single-page Ajax Interfaces. Proceedings of the 11th European Conference on Software Maintenance and Reengineering. 2006. P. 59–68. DOI: 10.1109/CSMR.2007.9.
12. Abramova V., Bernardino J. NoSQL Databases: MongoDB vs Cassandra. Proceedings of the International C* Conference on Computer Science and Software Engineering. 2013. P. 14–22. DOI: 10.1145/2494444.2494447.
13. Fielding R. Architectural Styles and the Design of Network-based Software Architectures : дис. ... д-ра філос. наук. University of California, Irvine,

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

2000. 162 p. URL: <https://ics.uci.edu/~fielding/pubs/dissertation/top.htm> (дата звернення: 29.05.2026).

14. React Documentation. Meta Platforms Inc. URL: <https://react.dev> (дата звернення: 06.05.2026).

15. Node.js Documentation. OpenJS Foundation. URL: <https://nodejs.org/docs/latest/api/> (дата звернення: 11.05.2026).

16. Express.js Documentation. OpenJS Foundation. URL: <https://expressjs.com> (дата звернення: 01.06.2026).

17. MongoDB Documentation. Mongo Inc. URL: <https://www.mongodb.com/docs/> (дата звернення: 17.05.2026).

18. Mongoose Documentation. Automattic Inc. URL: <https://mongoosejs.com> (дата звернення: 19.05.2026).

19. TypeScript Documentation. Microsoft Corporation. URL: <https://www.typescriptlang.org/docs/> (дата звернення: 16.05.2026).

20. Tailwind CSS Documentation. Tailwind Labs Inc. URL: <https://tailwindcss.com/docs/installation/using-vite> (дата звернення: 16.05.2026).

21. Cloudinary Documentation. Cloudinary Ltd. URL: <https://cloudinary.com/documentation> (дата звернення: 12.06.2026).

22. JSON Web Token RFC 7519. IETF. URL: <https://datatracker.ietf.org/doc/html/rfc7519> (дата звернення: 28.05.2026).

23. Nova Poshta API Documentation. Нова Пошта. URL: <https://developers.novaposhta.ua/documentation> (дата звернення: 13.06.2026).

24. Zustand Documentation. pmndrs. URL: <https://zustand.docs.pmnd.rs/learn/getting-started/introduction> (дата звернення: 09.06.2026).

25. Vite Documentation. Vite Team. URL: <https://vite.dev/guide/> (дата звернення: 03.05.2026).

26. React Router Documentation. Remix Software. URL: <https://reactrouter.com> (дата звернення: 05.05.2026).

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О.Г.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

27. Axios Documentation. Axios Contributors. URL:
<https://axios.rest/pages/getting-started/first-steps> (дата звернення: 27.05.2026).
28. Bcrypt.js. npm package documentation. URL:
<https://www.npmjs.com/package/bcryptjs> (дата звернення: 05.05.2026).
29. Multer. npm package documentation. URL:
<https://www.npmjs.com/package/multer> (дата звернення: 05.05.2026).
30. Lucide React Documentation. Lucide Contributors. URL:
<https://lucide.dev/> (дата звернення: 12.05.2026).
31. React Documentation. Components and Props. Meta Platforms Inc. URL:
<https://react.dev/learn/your-first-component> (дата звернення: 05.05.2026).
32. Using Middleware. Express.js Documentation. OpenJS Foundation. URL:
<https://expressjs.com/en/guide/using-middleware/> (дата звернення: 13.05.2026).

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О. Г.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмотря О.Г.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

Middleware авторизації:

```

import { Request, Response, NextFunction } from 'express'

import jwt from 'jsonwebtoken'
import User from '../models/User'

declare global {
  namespace Express {
    interface Request {
      user?: {
        id: string
        role: string
      }
    }
  }
}

export const protect = async (req: Request, res: Response, next: NextFunction):
Promise<void> => {
  try {
    const token = req.headers.authorization?.split(' ')[1]
    if (!token) {
      res.status(401).json({ message: 'Not authorized' })
      return
    }

    const decoded = jwt.verify(token, process.env.JWT_SECRET as string) as { id:
string }
    const user = await User.findById(decoded.id)
    if (!user) {
      res.status(401).json({ message: 'User not found' })
      return
    }

    req.user = { id: user._id.toString(), role: user.role }
    next()
  } catch {
    res.status(401).json({ message: 'Invalid token' })
  }
}

export const adminOnly = (req: Request, res: Response, next: NextFunction): void =>
{
  if (req.user?.role !== 'admin') {
    res.status(403).json({ message: 'Admin access required' })
    return
  }
  next()
}

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІЗп	Арк.
		Чижмоторя О.Г.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

Middleware завантаження зображень:

```
import { CloudinaryStorage } from 'multer-storage-cloudinary'
import multer from 'multer'
import cloudinary from '../config/cloudinary'

const storage = new CloudinaryStorage({
  cloudinary,
  params: {
    folder: 'caverty-mall',
    allowed_formats: ['jpg', 'jpeg', 'png', 'webp'],
    transformation: [{ width: 800, height: 1000, crop: 'limit' }],
  } as any,
})

export const upload = multer({ storage })
```

Контролер товарів:

```
import { Request, Response } from 'express'
import Product from '../models/Product'

export const getProducts = async (req: Request, res: Response): Promise<void> => {
  try {
    const {
      category, gender, size, color, brand,
      material, season, minPrice, maxPrice,
      search, sort, page = 1, limit = 12,
      isNew, isFeatured, minRating
    } = req.query

    const filter: any = { isActive: true }

    if (category) filter.category = category
    if (gender) filter.gender = gender
    if (brand) filter.brand = { $regex: brand, $options: 'i' }
    if (material) filter.material = material
    if (season) filter.season = { $in: Array.isArray(season) ? season : [season] }
    if (color) filter.colors = { $in: Array.isArray(color) ? color : [color] }
    if (isNew === 'true') filter.isNew = true
    if (isFeatured === 'true') filter.isFeatured = true
    if (search) filter.$text = { $search: search as string }
    if (minRating) filter.rating = { $gte: Number(minRating) }

    if (minPrice || maxPrice) {
      filter.price = {}
      if (minPrice) filter.price.$gte = Number(minPrice)
      if (maxPrice) filter.price.$lte = Number(maxPrice)
    }

    if (size) {
      filter.stock = {
```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    $elemMatch: {
      size: { $in: Array.isArray(size) ? size : [size] },
      quantity: { $gt: 0 }
    }
  }
}

const sortOptions: any = {
  'popular': { reviewCount: -1 },
  'rating': { rating: -1 },
  'price_asc': { price: 1 },
  'price_desc': { price: -1 },
  'newest': { createdAt: -1 },
  'name_asc': { name: 1 },
  'name_desc': { name: -1 },
}

const sortOption = sortOptions[sort as string] || { createdAt: -1 }

const total = await Product.countDocuments(filter)
const products = await Product.find(filter)
  .sort(sortOption)
  .skip((Number(page) - 1) * Number(limit))
  .limit(Number(limit))

res.json({
  success: true,
  data: products,
  pagination: {
    total,
    page: Number(page),
    pages: Math.ceil(total / Number(limit))
  }
})
} catch {
  res.status(500).json({ message: 'Server error' })
}
}

export const getProductById = async (req: Request, res: Response): Promise<void> =>
{
  try {
    const product = await Product.findOne({ _id: req.params.id, isActive: true })
    if (!product) {
      res.status(404).json({ message: 'Product not found' })
      return
    }
    res.json({ success: true, data: product })
  } catch {
    res.status(500).json({ message: 'Server error' })
  }
}

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

export const createProduct = async (req: Request, res: Response): Promise<void> =>
{
  try {
    const files = req.files as Express.Multer.File[]

    const images = files?.length
      ? files.map((file: any) => ({
          url: file.path,
          publicId: file.filename,
        })))
      : []

    let stock = []
    try {
      stock = typeof req.body.stock === 'string'
        ? JSON.parse(req.body.stock)
        : req.body.stock || []
    } catch {
      stock = []
    }

    let sizes = []
    try {
      sizes = typeof req.body.sizes === 'string'
        ? JSON.parse(req.body.sizes)
        : req.body.sizes || []
    } catch {
      sizes = []
    }

    let colors = []
    try {
      colors = typeof req.body.colors === 'string'
        ? JSON.parse(req.body.colors)
        : req.body.colors || []
    } catch {
      colors = []
    }

    let season = []
    try {
      season = typeof req.body.season === 'string'
        ? JSON.parse(req.body.season)
        : req.body.season || []
    } catch {
      season = []
    }

    const product = await Product.create({

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІЗп	Арк.
		Чижмоторя О.Г.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        name: req.body.name,
        description: req.body.description,
        price: Number(req.body.price),
        category: req.body.category,
        gender: req.body.gender || 'unisex',
        brand: req.body.brand || '',
        material: req.body.material || '',
        season: season,
        colors: colors,
        sizes: sizes,
        stock: stock,
        images: images,
        isActive: req.body.isActive === 'true',
        isNew: req.body.isNew === 'true',
        isFeatured: req.body.isFeatured === 'true',
    })

    res.status(201).json({ success: true, data: product })
  } catch (error: any) {
    res.status(400).json({ message: error.message })
  }
}

export const updateProduct = async (req: Request, res: Response): Promise<void> =>
{
  try {
    const files = req.files as Express.Multer.File[]

    let stock = []
    try {
      stock = typeof req.body.stock === 'string'
        ? JSON.parse(req.body.stock)
        : req.body.stock || []
    } catch {
      stock = []
    }

    let sizes = []
    try {
      sizes = typeof req.body.sizes === 'string'
        ? JSON.parse(req.body.sizes)
        : req.body.sizes || []
    } catch {
      sizes = []
    }

    let colors = []
    try {
      colors = typeof req.body.colors === 'string'
        ? JSON.parse(req.body.colors)
        : req.body.colors || []
    }
  }
}

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІЗп	Арк.
		Чижмоторя О.Г.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    } catch {
      colors = []
    }

    let season = []
    try {
      season = typeof req.body.season === 'string'
        ? JSON.parse(req.body.season)
        : req.body.season || []
    } catch {
      season = []
    }

    const updateData: any = {
      name: req.body.name,
      description: req.body.description,
      price: Number(req.body.price),
      category: req.body.category,
      gender: req.body.gender || 'unisex',
      brand: req.body.brand || '',
      material: req.body.material || '',
      season: season,
      colors: colors,
      sizes: sizes,
      stock: stock,
      isActive: req.body.isActive === 'true',
      isNew: req.body.isNew === 'true',
      isFeatured: req.body.isFeatured === 'true',
    }

    if (files?.length) {
      updateData.images = files.map((file: any) => ({
        url: file.path,
        publicId: file.filename,
      })))
    }

    const product = await Product.findByIdAndUpdate(
      req.params.id,
      updateData,
      { new: true, runValidators: true }
    )
    if (!product) {
      res.status(404).json({ message: 'Product not found' })
      return
    }
    res.json({ success: true, data: product })
  } catch (error: any) {
    res.status(400).json({ message: error.message })
  }
}

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

export const deleteProduct = async (req: Request, res: Response): Promise<void> =>
{
  try {
    const product = await Product.findByIdAndUpdate(
      req.params.id,
      { isActive: false },
      { new: true }
    )
    if (!product) {
      res.status(404).json({ message: 'Product not found' })
      return
    }
    res.json({ success: true, message: 'Product deleted' })
  } catch {
    res.status(500).json({ message: 'Server error' })
  }
}

```

Контролер відгуків:

```

import { Request, Response } from 'express'
import Review from '../models/Review'
import Product from '../models/Product'

export const getReviews = async (req: Request, res: Response): Promise<void> => {
  try {
    const productId = req.params.productId as string
    const reviews = await Review.find({ productId })
      .sort({ createdAt: -1 })
    res.json({ success: true, data: reviews })
  } catch {
    res.status(500).json({ message: 'Server error' })
  }
}

export const createReview = async (req: Request, res: Response): Promise<void> => {
  try {
    const { rating, comment } = req.body
    const productId = req.params.productId as string

    const existing = await Review.findOne({
      productId,
      userId: req.user?.id,
    })
    if (existing) {
      res.status(400).json({ message: 'Ви вже залишили відгук на цей товар' })
      return
    }

    const review = await Review.create({

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        productId,
        userId: req.user?.id,
        userName: req.body.userName,
        rating,
        comment,
    })

    const reviews = await Review.find({ productId })
    const avgRating = reviews.reduce((sum, r) => sum + r.rating, 0) /
reviews.length
    await Product.findByIdAndUpdate(productId, {
        rating: Math.round(avgRating * 10) / 10,
        reviewCount: reviews.length,
    })

    res.status(201).json({ success: true, data: review })
} catch (error: any) {
    if (error.code === 11000) {
        res.status(400).json({ message: 'Ви вже залишили відгук на цей товар' })
        return
    }
    res.status(400).json({ message: error.message })
}
}

export const deleteReview = async (req: Request, res: Response): Promise<void> => {
    try {
        const review = await Review.findById(req.params.reviewId)
        if (!review) {
            res.status(404).json({ message: 'Відгук не знайдено' })
            return
        }

        if (review.userId.toString() !== req.user?.id && req.user?.role !== 'admin') {
            res.status(403).json({ message: 'Немає прав' })
            return
        }

        await Review.findByIdAndDelete(req.params.reviewId)

        const reviews = await Review.find({ productId: review.productId })
        const avgRating = reviews.length
        ? reviews.reduce((sum, r) => sum + r.rating, 0) / reviews.length
        : 0
        await Product.findByIdAndUpdate(review.productId, {
            rating: Math.round(avgRating * 10) / 10,
            reviewCount: reviews.length,
        })

        res.json({ success: true, message: 'Відгук видалено' })
    }
}

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    } catch {
      res.status(500).json({ message: 'Server error' })
    }
  }
  export const replyToReview = async (req: Request, res: Response): Promise<void>
=> {
  try {
    const { comment } = req.body
    const review = await Review.findByIdAndUpdate(
      req.params.reviewId,
      {
        $push: {
          replies: {
            userId: req.user?.id,
            userName: req.body.userName,
            comment,
          }
        }
      },
      { new: true }
    )
    if (!review) {
      res.status(404).json({ message: 'Відгук не знайдено' })
      return
    }
    res.json({ success: true, data: review })
  } catch {
    res.status(500).json({ message: 'Server error' })
  }
}

export const editReview = async (req: Request, res: Response): Promise<void> => {
  try {
    const review = await Review.findById(req.params.reviewId)
    if (!review) {
      res.status(404).json({ message: 'Відгук не знайдено' })
      return
    }

    if (review.userId.toString() !== req.user?.id && req.user?.role !== 'admin') {
      res.status(403).json({ message: 'Немає прав' })
      return
    }

    review.rating = req.body.rating ?? review.rating
    review.comment = req.body.comment ?? review.comment
    await review.save()

    const reviews = await Review.find({ productId: review.productId })
    const avgRating = reviews.reduce((sum, r) => sum + r.rating, 0) /
    reviews.length
  }
}

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    await Product.findByIdAndUpdate(review.productId, {
      rating: Math.round(avgRating * 10) / 10,
    })

    res.json({ success: true, data: review })
  } catch {
    res.status(500).json({ message: 'Server error' })
  }
}

export const editReply = async (req: Request, res: Response): Promise<void> => {
  try {
    const review = await Review.findById(req.params.reviewId)
    if (!review) {
      res.status(404).json({ message: 'Відгук не знайдено' })
      return
    }

    const reply = review.replies.find(
      (r) => r._id.toString() === req.params.replyId
    )

    if (!reply) {
      res.status(404).json({ message: 'Відповідь не знайдено' })
      return
    }

    if (reply.userId.toString() !== req.user?.id && req.user?.role !== 'admin') {
      res.status(403).json({ message: 'Немає прав' })
      return
    }

    reply.comment = req.body.comment
    await review.save()

    res.json({ success: true, data: review })
  } catch {
    res.status(500).json({ message: 'Server error' })
  }
}

export const deleteReply = async (req: Request, res: Response): Promise<void> => {
  try {
    const review = await Review.findById(req.params.reviewId)
    if (!review) {
      res.status(404).json({ message: 'Відгук не знайдено' })
      return
    }

    const reply = review.replies.find(
      (r) => r._id.toString() === req.params.replyId
    )
  }
}

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІЗп	Арк.
		Чижмоторя О.Г.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    )
    if (!reply) {
      res.status(404).json({ message: 'Відповідь не знайдено' })
      return
    }

    if (reply.userId.toString() !== req.user?.id && req.user?.role !== 'admin') {
      res.status(403).json({ message: 'Немає прав' })
      return
    }

    review.replies = review.replies.filter(
      (r) => r._id.toString() !== req.params.replyId
    ) as any

    await review.save()
    res.json({ success: true, message: 'Відповідь видалено' })
  } catch {
    res.status(500).json({ message: 'Server error' })
  }
}

```

Контролер замовлень:

```

import { Request, Response } from 'express'
import Order from '../models/Order'

export const createOrder = async (req: Request, res: Response): Promise<void> => {
  try {
    const { items, delivery, payment, totalPrice } = req.body

    if (!items || items.length === 0) {
      res.status(400).json({ message: 'Кошик порожній' })
      return
    }

    const order = await Order.create({
      userId: req.user?.id,
      items,
      delivery,
      payment,
      totalPrice,
    })

    res.status(201).json({ success: true, data: order })
  } catch (error: any) {
    res.status(400).json({ message: error.message })
  }
}

export const getMyOrders = async (req: Request, res: Response): Promise<void> => {
  try {

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІІЗп	Арк.
		Чижмоторя О.Г.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    const orders = await Order.find({ userId: req.user?.id }).sort({ createdAt: -1
  })
  res.json({ success: true, data: orders })
} catch {
  res.status(500).json({ message: 'Server error' })
}
}

export const getAllOrders = async (req: Request, res: Response): Promise<void> => {
  try {
    const orders = await Order.find().populate('userId', 'name email').sort({
      createdAt: -1 })
    res.json({ success: true, data: orders })
  } catch {
    res.status(500).json({ message: 'Server error' })
  }
}

export const cancelOrder = async (req: Request, res: Response): Promise<void> => {
  try {
    const order = await Order.findById(req.params.id)
    if (!order) {
      res.status(404).json({ message: 'Замовлення не знайдено' })
      return
    }

    if (order.userId.toString() !== req.user?.id) {
      res.status(403).json({ message: 'Немає прав' })
      return
    }

    if (order.status !== 'pending') {
      res.status(400).json({ message: 'Можна скасувати лише замовлення зі статусом
"Очікує"' })
      return
    }

    order.status = 'cancelled'
    await order.save()

    res.json({ success: true, data: order })
  } catch {
    res.status(500).json({ message: 'Server error' })
  }
}

export const updateOrderStatus = async (req: Request, res: Response): Promise<void>
=> {
  try {
    const { status } = req.body

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІІЗп	Арк.
		Чижмоторя О.Г.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    const validStatuses = ['pending', 'processing', 'shipped', 'delivered',
'cancelled', 'completed']
    if (!validStatuses.includes(status)) {
        res.status(400).json({ message: 'Невірний статус' })
        return
    }
    const order = await Order.findByIdAndUpdate(
        req.params.id,
        { status },
        { new: true }
    )
    if (!order) {
        res.status(404).json({ message: 'Замовлення не знайдено' })
        return
    }
    res.json({ success: true, data: order })
} catch {
    res.status(500).json({ message: 'Server error' })
}
}

```

		Мірецький Вл.Я.			ДУ «Житомирська політехніка».26.121.000 - ІПЗп	Арк.
		Чижмоторя О. Г.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

Сховище кошика

```

import { create } from 'zustand'
import { persist } from 'zustand/middleware'
import type { CartItem, Product } from '../types'

interface CartState {
  items: CartItem[]
  isOpen: boolean
  openCart: () => void
  closeCart: () => void
  toggleCart: () => void
  addItem: (product: Product, size: string, quantity?: number) => void
  removeItem: (productId: string, size: string) => void
  updateQuantity: (productId: string, size: string, quantity: number) => void
  clearCart: () => void
  totalItems: () => number
  totalPrice: () => number
}

export const useCartStore = create<CartState>()(
  persist(
    (set, get) => ({
      items: [],
      isOpen: false,

      openCart: () => set({ isOpen: true }),
      closeCart: () => set({ isOpen: false }),
      toggleCart: () => set((state) => ({ isOpen: !state.isOpen })),

      addItem: (product, size, quantity = 1) => {
        const existing = get().items.find(
          (i) => i.product._id === product._id && i.size === size
        )
        if (existing) {
          set((state) => ({
            items: state.items.map((i) =>
              i.product._id === product._id && i.size === size
                ? { ...i, quantity: i.quantity + quantity }
                : i
            ),
            isOpen: true,
          }))
        } else {
          set((state) => ({
            items: [...state.items, { product, size, quantity }],
            isOpen: true,
          }))
        }
      }
    })
  )
)

```

```

    },
    removeItem: (productId, size) => {
      set((state) => ({
        items: state.items.filter(
          (i) => !(i.product._id === productId && i.size === size)
        ),
      })))
    },
    updateQuantity: (productId, size, quantity) => {
      if (quantity <= 0) {
        get().removeItem(productId, size)
        return
      }
      set((state) => ({
        items: state.items.map((i) =>
          i.product._id === productId && i.size === size
            ? { ...i, quantity }
            : i
        ),
      })))
    },
    clearCart: () => set({ items: [] }),
    totalItems: () => get().items.reduce((sum, i) => sum + i.quantity, 0),
    totalPrice: () =>
      get().items.reduce((sum, i) => sum + i.product.price * i.quantity, 0),
  })),
{
  name: 'cart-storage',
  partialize: (state) => ({ items: state.items }),
}
)
)

```