

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 ювелірних прикрас «JEWELRY
LOUNGE»»**

Виконала студентка групи ВТ-21-1
ГАВРИЩУК Вікторія Олексіївна

Керівник роботи:
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:
ТОЛСТОЙ Ігор Анатолійович

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК

«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ГАВРИЩУК Вікторія Олексіївна**

Керівник роботи: **ЧИЖМОТРЯ Олена Геннадіївна**

Тема роботи: **«Інтернет-магазин ювелірних прикрас «JEWELRY LOUNGE»»,**

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: **Об'єктом дослідження є інтернет-магазин ювелірних прикрас «JEWELRY LOUNGE», призначений для забезпечення зручної та безпечної онлайн-торгівлі ювелірною продукцією. Предмет дослідження є процеси проєктування, розробки та впровадження функціональної веб-платформи для електронної комерції у сфері продажу ювелірних прикрас.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О. Г.	27.02.2025р.	27.02.2025р.
2	Чижмотря О. Г.	06.03.2025р.	06.03.2025р.
3	Чижмотря О. Г.	24.04.2025р.	24.04.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного комплексу та пояснювальної записки. Пояснювальна записка до випускної роботи містить 75 сторінки, включає 15 ілюстрації, 20 таблиць, 1 додаток та 30 використаних літературних джерел.

Метою роботи є створення інтернет-магазину для продажу ювелірних прикрас «JEWELRY LOUNGE» з використанням сучасних технологій. Система включає модулі для управління товарами, категоріями, замовленнями та користувачами, а також відображення змін у реальному часі.

У роботі поставлені основні завдання на розробку системи. Також було проведено порівняльний аналіз аналогічних систем. Було обрано монолітну архітектуру з використанням Laravel та MariaDB. Приведені основні сценарії використання системи, логіка взаємодії між модулями системи та алгоритми роботи ключових модулів, спроектована база даних з використанням MariaDB. Клієнтська та серверна частина розроблена за допомогою фреймворку Laravel.

Ключові слова: ІНТЕРНЕТ-МАГАЗИН, КОРИСТУВАЧ, БАЗА ДАНИХ, ПРОДУКТ, ПЛАТІЖНА СИСТЕМА.

					ІІЗ.КР.Б – 121 – 25 - ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	Інтернет-магазин ювелірних прикрас «JEWELRY LOUNGE» Пояснювальна записка	Літ.	Арк.	Аркушів	
Розроб.		Гавришук В.О.							
Керівник.		Чижмотря О.Г					3	75	
Рецензент		Толстой І.А.							
Зав. каф.		Вакалюк Т.А				Житомирська політехніка, група ВТ-21-1			

ABSTRACT

The bachelor's thesis consists of a software package and an explanatory note. The explanatory note to the graduation thesis is 75 pages long, includes 15 illustrations, 20 tables, 1 appendix and 30 references.

The goal of the work is to create an online store for selling jewelry, "JEWELRY LOUNGE," using modern technologies. The system includes modules for managing products, categories, orders, and users, as well as real-time updates.

The work outlines the main tasks for system development. A comparative analysis of similar systems was also conducted. A monolithic architecture was chosen, utilizing Laravel and MariaDB. The key use cases of the system, the logic of interaction between system modules, and the algorithms for key modules are presented, along with a database design using MariaDB. Both the client-side and server-side parts are developed using the Laravel framework.

Keywords: ONLINE STORE, USER, DATABASE, PRODUCT, PAYMENT SYSTEM.

					ПЗ.КР.Б – 121 – 25 - ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ІНТЕРНЕТ-МАГАЗИНУ ЮВЕЛІРНИХ ПРИКРАС.....	9
1.1. Визначення та постановка задачі	9
1.2. Аналіз аналогів програмного продукту	10
1.3. Визначення архітектури програмного забезпечення.....	13
1.4. Обґрунтування вибору інструментальних засобів та вимоги до програмного забезпечення	15
Висновки до першого розділу	19
РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ЮВЕЛІРНИХ ПРИКРАС.....	20
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	20
2.2. Розробка бази даних системи.....	31
2.3. Проєктування та реалізація алгоритмів роботи системи	39
Висновки до другого розділу	44
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІНТЕРНЕТ-МАГАЗИНУ ЮВЕЛІРНИХ ПРИКРАС	46
3.1. Порядок встановлення та налаштування параметрів системи	46
3.2. Структура інтерфейсу	47
3.3. Тестування роботи програмного комплексу	53
Висновки до третього розділу.....	56
ВИСНОВКИ.....	57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	58

					ІПЗ.КР.Б – 121 – 25 - ПЗ	Арк.
						5
Змн.	Арк.	№ докум.	Підпис	Дата		

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ - програмне забезпечення.

БД - база даних.

СУБД - система управління базами даних.

CRUD - create, read, update, delete.

PHP - Hypertext Preprocessor.

SQL - Structured Query Language.

HTML - HyperText Markup Language.

CSS - Cascading Style Sheets.

JS - JavaScript.

					ПЗ.КР.Б – 121 – 25 - ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми пов'язана зі зростанням популярності інтернет-магазинів, зокрема ювелірних прикрас, що ставить перед командами завдання у вигляді ефективного управління процесами продажу, клієнтського обслуговування та логістики. В умовах високої конкуренції на ринку онлайн-продажів виникає потреба в автоматизації та вдосконаленні бізнес-процесів. Розробка інтернет-магазину для ювелірних прикрас, що використовує сучасні технології для організації продажів, управління товарами, замовленнями та клієнтами, дозволяє підвищити ефективність роботи магазину, покращити досвід покупців та забезпечити успіх на ринку.

Метою випускної кваліфікаційної роботи бакалавра є розробка інтернет-магазину ювелірних прикрас «JEWELRY LOUNGE» з використанням сучасних технологій веб-розробки, що забезпечують ефективне управління товарами, замовленнями та платежами, а також покращують користувацький досвід.

Для реалізації поставленої мети потрібно виконати наступні **завдання**:

- Провести огляд існуючих рішень та платформ для онлайн-продажу ювелірних прикрас;
- Обрати архітектуру та інструментальні засоби для розробки інтернет-магазину;
- Спроектувати систему управління товарами, замовленнями та платежами;
- Розробити інтернет-магазин для продажу ювелірних прикрас з інтеграцією платіжних систем.

Об'єктом дослідження є інтернет-магазин ювелірних прикрас, що призначений для організації процесів онлайн-продажу, управління товарами, замовленнями та платежами, зокрема в сфері ювелірних прикрас.

Предметом дослідження є процеси проектування, розробки та реалізації інтернет-магазину, включаючи взаємодію з платіжними системами та оптимізацію взаємодії користувачів з платформою.

					ІПЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ІНТЕРНЕТ-МАГАЗИНУ ЮВЕЛІРНИХ ПРИКРАС

1.1. Визначення та постановка задачі

Метою випускної кваліфікаційної роботи бакалавра є розробка інтернет-магазину ювелірних прикрас «JEWELRY LOUNGE» з використанням сучасних технологій для ефективного управління товарами, замовленнями, користувачами та інтеграцією платіжних систем.

Інтернет-магазини для продажу ювелірних прикрас стають все більш популярними, і відповідно зростає потреба в автоматизації всіх бізнес-процесів. Для того, щоб забезпечити безперебійний процес покупки, необхідно створити систему, яка дозволить користувачам легко знаходити товари, оформляти замовлення та здійснювати оплату через безпечні платіжні системи.

Платформа для інтернет-магазину «JEWELRY LOUNGE» складається з кількох ключових модулів. Модуль товарів забезпечує можливість створення, редагування та управління товарами, включаючи їх категорії, ціни, зображення та опис. Модуль користувачів дозволяє реєструвати користувачів, керувати їх профілями та правами доступу до різних функцій системи. Модуль замовлень служить основним інструментом для створення, управління та відстеження замовлень з можливістю редагування статусу та відправки сповіщень. Модуль платіжних систем інтегрується з платіжними системами, такими як LiqPay [6], для обробки онлайн-платежів. Модуль реального часу використовує WebSockets для відображення оновлень замовлень та статусів у реальному часі, що дозволяє клієнтам отримувати актуальну інформацію без необхідності перезавантаження сторінки. Модуль сповіщень надсилає сповіщення користувачам про зміни в замовленнях, нові акції, оновлення товарів через електронну пошту та push-сповіщення.

Розробка інтернет-магазину передбачає кілька етапів. Спочатку потрібно спроектувати загальну структуру додатку, визначити основну концепцію

					ІПЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

платформи та її ключові функціональні можливості, а також спланувати логіку взаємодії між користувачами, проєктами та робочими елементами. Після цього необхідно розробити модель даних, що описує сутності та їхні зв'язки. На наступному етапі важливо побудувати архітектуру клієнтської та серверної частини системи, визначити загальну архітектуру, планувати способи обробки запитів та зберігання даних на сервері. Далі потрібно реалізувати базовий функціонал, включаючи систему аутентифікації та авторизації користувачів, управління товарами та замовленнями, а також механізм відображення оновлень даних у реальному часі. Останнім етапом є тестування та налагодження системи, де потрібно провести функціональне тестування для перевірки відповідності розробленого функціоналу вимогам, а також виявити та усунути програмні помилки.

Результатом реалізації поставленого завдання стане інтернет-магазин для продажу ювелірних прикрас «JEWELRY LOUNGE», який дозволяє покупцям зручно переглядати товари, оформляти замовлення, здійснювати оплату та отримувати сповіщення про статуси своїх замовлень у реальному часі.

1.2. Аналіз аналогів програмного продукту

У сучасному цифровому просторі існує безліч інтернет-магазинів, що пропонують ювелірні вироби. Для визначення оптимальних рішень та функціональних можливостей майбутньої платформи "JEWELRY LOUNGE" розглянемо декілька популярних онлайн-магазинів ювелірних прикрас. Ось деякі з них.

"Золотий Вік" - це один з найбільших ювелірних магазинів в Україні, що пропонує прикраси з різних видів золота та срібла (рис. 1.1). Сайт надає інформацію про акції, програми лояльності та гарантійне обслуговування. Однак функціональність сайту обмежується стандартними можливостями інтернет-магазину без інтерактивних елементів або персоналізації.

Завдяки широкому асортименту товарів, зручному інтерфейсу та можливості оформлення доставки по всій Україні, "Золотий Вік" займає провідні позиції на

					ІПЗ.КР.Б – 121 – 25 - ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

ринку ювелірних прикрас. Однак сайт не має достатньо інтеграцій з соціальними мережами або додаткових інструментів для взаємодії з користувачами, що обмежує можливості для створення більш персоналізованого досвіду покупки.

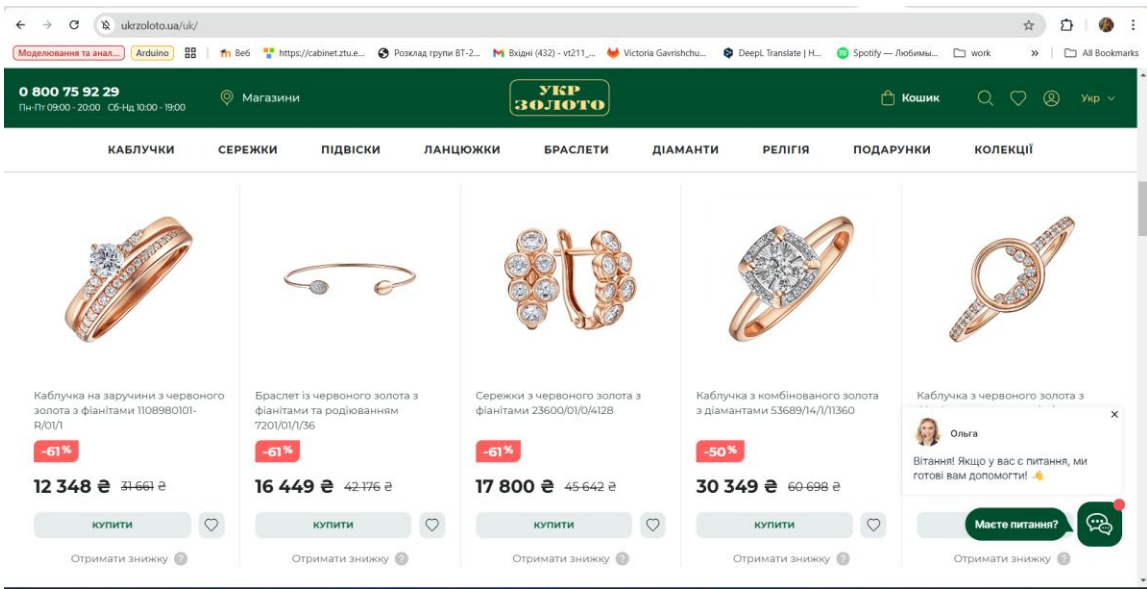


Рис. 1.1 - Головна сторінка інтернет-магазину "Золотий Вік"

AURUM - ювелірний магазин, що пропонує прикраси з золота та срібла, а також аксесуари й подарункові сертифікати. Сайт має сучасний дизайн (рис. 1.2), містить розділ із часто задаваними питаннями, а також пропонує програми лояльності та онлайн-замовлення. Однак функціональність пошуку товарів обмежена, оскільки відсутні розширені фільтри або рекомендаційна система.

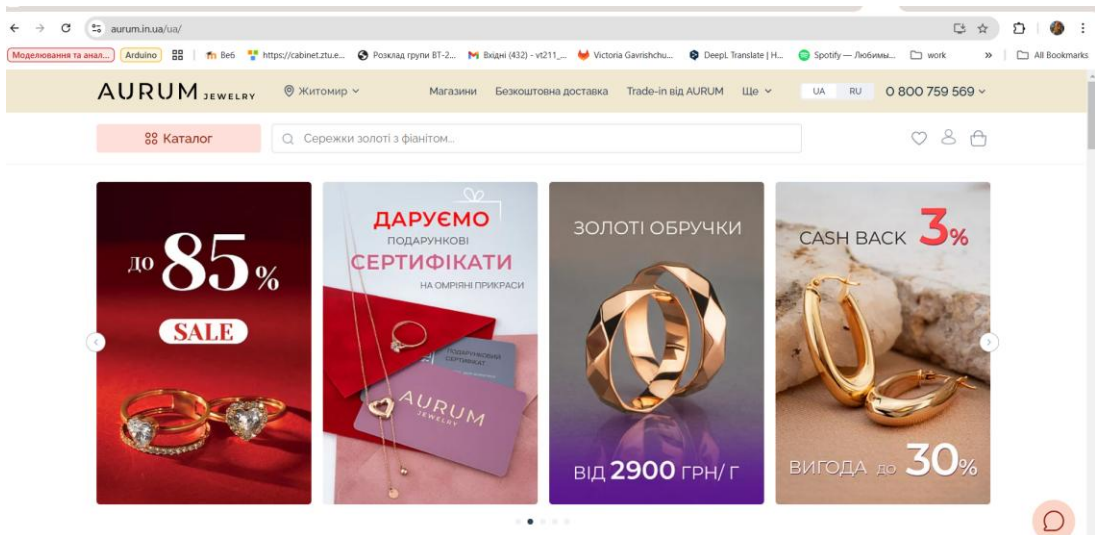


Рис. 1.2 - Головна сторінка інтернет-магазину AURUM

"Brilliant Earth" є американським інтернет-магазином, який спеціалізується на етичних та екологічних практиках, особливо щодо вирощених у лабораторії діамантів (рис. 1.3). Платформа пропонує можливість кастомізації прикрас, розширений пошук та інтерактивні елементи, але для українських користувачів можуть виникати складнощі з доставкою та митними процедурами.

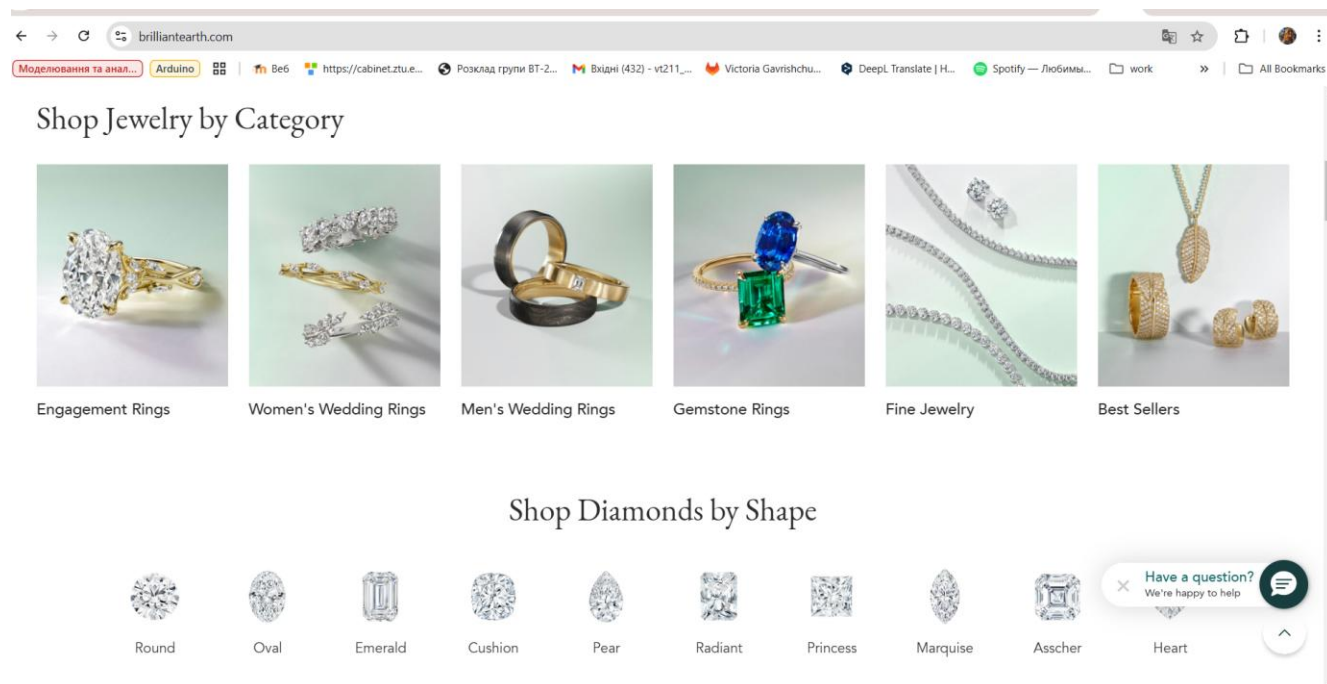


Рис. 1.3 - Головна сторінка інтернет-магазину Brilliant Earth

Виходячи з наявних переваг та недоліків вищезгаданих аналогів, складемо порівняльну таблицю їх функціональних можливостей (табл. 1.1).

Таблиця 1.1

Функціональні можливості

Функціонал	Золотий Вік	AURUM	Brilliant Earth
Асортимент	Різноманітні прикраси з різних видів золота та срібла	Прикраси з золота, срібла, аксесуари	Дизайнерські прикраси з етичних матеріалів
Інтерфейс сайту	Інтуїтивно зрозумілий, інформація про акції	Сучасний дизайн, розділ FAQ	Сучасний, з акцентом на етичність
Можливість кастомізації	Відсутня	Відсутня	Обмежена

Додаткові сервіси	Програми лояльності, гарантійне обслуговування	Подарункові сертифікати, програма лояльності	Етичні діаманти, екологічні практики
Інтерактивні елементи	Відсутні	Відсутні	Відсутні
Цільова аудиторія	Широке коло споживачів	Широке коло споживачів	Споживачі, зацікавлені в етичних прикрасах

Кожна з розглянутих платформ має свої особливості та унікальні можливості, які підходять для різних команд і типів проєктів. "Золотий Вік" пропонує широкий асортимент ювелірних прикрас, але не має інтерактивних елементів. "AURUM" має сучасний дизайн та пропонує програми лояльності, але теж не надає користувачам можливості кастомізації. "Brilliant Earth" - це платформа, яка використовує етичні матеріали та пропонує інтерактивні можливості для персоналізації, таких як кастомізація прикрас і віртуальна примірка, але має високу вартість та обмежену доступність для українського ринку.

Платформа для інтернет-магазину "JEWELRY LOUNGE" повинна включати такі функції, які дозволяють користувачам не тільки комфортно купувати ювелірні вироби, а й мати можливість налаштувати продукти під себе. Це включає можливість кастомізації, інтерактивні елементи, персоналізовані рекомендації та зручні сервіси для покупців.

В результаті, порівняння допомогло визначити важливі функції, які мають бути інтегровані у платформу "JEWELRY LOUNGE", щоб вона могла конкурувати з іншими лідерами ринку.

1.3. Визначення архітектури програмного забезпечення

Для розробки онлайн-магазину сучасних ювелірних прикрас «JEWELRY LOUNGE» було обрано клієнт-серверну архітектуру, де клієнтська частина

взаємодіє з сервером для отримання, обробки та збереження даних (рис. 1.4). Такий підхід забезпечує масштабованість, модульність та можливість розширення функціональності.

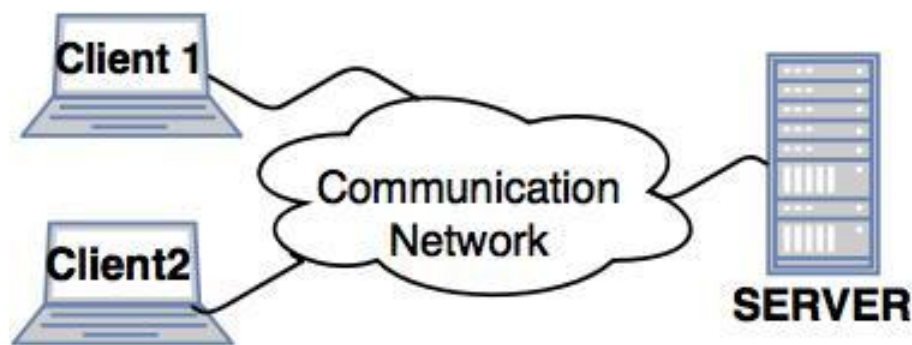


Рис. 1.4 - Схематичне зображення клієнтсько-серверної архітектури

Клієнтська частина реалізована за допомогою Blade шаблонів у Laravel [1], що відповідає за інтерфейс користувача, перегляд каталогу товарів, оформлення замовлень та взаємодію із сервером через REST API. Серверна частина побудована на PHP Laravel, який відповідає за бізнес-логіку, обробку замовлень, управління користувачами та інтеграцію з платіжними системами [2]. База даних реалізована на MariaDB, що гарантує надійність збереження даних та їх швидку обробку [3].

Laravel використовує шаблон MVC (Model-View-Controller), що дозволяє ефективно організувати структуру програми, розділяючи логіку обробки даних, представлення та взаємодії з користувачем. Це дозволяє спростити підтримку коду та покращити тестованість програми. Крім того, Laravel включає елементи інверсії керування (IoC), що дає можливість знижувати зв'язність між компонентами системи та покращувати гнучкість при розробці нових функцій. Залежності між компонентами управляються через сервіс-контейнер, що дозволяє легко налаштовувати сервіси та підвищує тестованість.

Також у реалізації використовуються такі патерни проектування, як Singleton і Factory. Патерн Singleton застосовується для створення одного екземпляра сервісів, таких як підключення до бази даних або авторизація користувачів, щоб забезпечити єдину точку доступу до цих служб в межах програми [4]. Патерн Factory використовується для створення об'єктів, пов'язаних з різними типами

продуктів та категоріями, що дозволяє абстрагувати логіку створення об'єктів та спрощує її тестування та модифікацію [5].

Серверна частина реалізує бізнес-логіку та інтеграцію з різними сервісами. Вона включає обробку запитів через REST API, що дозволяє легко взаємодіяти з клієнтською частиною, а також реалізує сповіщення у реальному часі через WebSockets для оперативного оновлення інформації (наприклад, для миттєвих сповіщень та оновлень контенту).

Ця архітектура дозволяє ефективно масштабувати проєкт, додаючи нові функціональні модулі або покращуючи наявні. Завдяки модульності та чітко розділеній логіці клієнтської та серверної частин система легко обслуговується та розвивається без значних переробок існуючої структури. Оскільки кожен компонент можна розробляти та тестувати окремо, це забезпечує високу продуктивність та мінімізацію можливих помилок при розробці та запуску нових можливостей.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до програмного забезпечення

Для розробки онлайн-магазину сучасних ювелірних прикрас «JEWELRY LOUNGE» було обрано інструментальні засоби, що відповідають вимогам високої продуктивності, масштабованості та зручності в обслуговуванні. Вибір технологій був обумовлений необхідністю забезпечити ефективне управління товарними позиціями, обробку замовлень та інтеграцію з платіжними системами.

Для реалізації серверної частини було обрано фреймворк Laravel, оскільки він надає зручні інструменти для розробки масштабованих веб-застосунків із чітко організованою структурою. Laravel також включає функціональність для реалізації безпечного зберігання даних та роботи з базою даних. Для зберігання даних використано MariaDB, яка забезпечує високу надійність, швидкість обробки запитів та оптимальну підтримку реляційної моделі даних.

Для побудови інтерактивного інтерфейсу було обрано Blade шаблони, які дозволяють організувати рендеринг сторінок у Laravel. Вони забезпечують

					ІПЗ.КР.Б – 121 – 25 - ІЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

гнучкість у створенні динамічних інтерфейсів, а також підтримують взаємодію з сервером через REST API. RESTful архітектура дозволяє забезпечити ефективну комунікацію між клієнтською та серверною частинами, дозволяючи користувачам здійснювати запити до системи для оформлення замовлень і перегляду каталогу товарів.

Також для взаємодії з платіжними системами було інтегровано LiqPay, яка надає зручний інтерфейс для здійснення платежів через інтернет. Вона забезпечує безпечну передачу даних та зручну інтеграцію з веб-сервісами.

У результаті вибору цих інструментів було отримано систему, яка забезпечує високий рівень безпеки даних, ефективне управління товарами, обробку замовлень та інтеграцію з платіжними системами. Вибір Laravel, MariaDB та LiqPay дозволяє зберегти гнучкість і масштабованість платформи, що є важливим для її подальшого розвитку та розширення.

Для забезпечення високої продуктивності системи було використано принципи розподілу навантаження через налаштування кешування та оптимізації запитів до бази даних. Крім того, при виборі інструментів було враховано їх популярність та активну підтримку спільнотою розробників, що дозволяє забезпечити стабільну роботу системи в умовах змінних вимог.

Також для забезпечення зручності користувачів була інтегрована система адміністрування, що дозволяє легко керувати товарами, категоріями та замовленнями. Це дає змогу адміністраторам швидко додавати нові продукти, змінювати їхні параметри та відслідковувати поточні замовлення через зручний інтерфейс. Використання Laravel дозволяє забезпечити швидкий розвиток додаткових функціональних можливостей і підтримку нових платіжних систем або інтеграцій за потреби.

Виходячи з наявних переваг та недоліків обраних інструментів для розробки інтернет-магазину "JEWELRY LOUNGE", було складено порівняльну таблицю їх функціональних можливостей, яка дозволяє оцінити різні аспекти кожного інструменту та їх взаємодію в рамках загальної архітектури проєкту (табл. 1.2).

					ІПЗ.КР.Б – 121 – 25 - ІПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

Порівняння інструментів

Критерій	Laravel	MariaDB	LiqPay
Продуктивність	Висока	Висока	Висока
Масштабованість	Висока	Висока	Середня
Простота використання	Середня	Середня	Висока
Безпека	Висока	Висока	Висока
Інтеграція з іншими системами	Висока	Висока	Висока

Ось такий вибір інструментів забезпечує стабільну та ефективну роботу інтернет-магазину «JEWELRY LOUNGE» з можливістю його подальшого розвитку. Завдяки використанню перевірених технологій, таких як Laravel, MariaDB та LiqPay, платформа здатна забезпечити безпечну та швидку обробку замовлень, що є критично важливим для успішного функціонування інтернет-магазину в сучасних умовах.

Також створимо таблицю, що порівнює MariaDB з іншими популярними СУБД [26], і вказує, чому саме MariaDB була обрана для нашого інтернет-магазину ювелірних прикрас (табл. 1.3), а не будь-яку іншу, наприклад MongoDB [29] чи PostgreSQL [28].

Порівняння баз даних дозволяє зрозуміти переваги кожної технології в контексті використання в інтернет-магазинах, де важливими є швидкість роботи з даними, можливості масштабування та надійність системи в умовах зростаючих навантажень. Ми вибрали MariaDB, оскільки вона поєднує переваги реляційних баз даних з високою продуктивністю і надійною безпекою, що важливо для нашої платформи.

Порівняння баз даних для використання в проєкті

Критерій	MongoDB	PostgreSQL	MariaDB
Тип СУБД	Документна (NoSQL)	Реляційна (SQL)	Реляційна (SQL)
Структура даних	JSON-документи	Таблична, підтримка JSONB	Таблична, базова підтримка JSON
Масштабованість	Горизонтальна	Горизонтальна і вертикальна	Вертикальна (стабільна), з можливістю шардінгу
Продуктивність на складних запитах	Висока на неструктурованих даних	Висока продуктивність на складних аналітичних запитах	Висока продуктивність для більшості CRUD-операцій
Робота з великими обсягами даних	Оптимізована для великих неструктурованих обсягів	Сильна підтримка великих масивів даних	Надійна при помірному навантаженні, добре масштабується вертикально
Підтримка JSON	Основний формат зберігання	Так (JSONB - оптимізований для запитів та зберігання)	Так (є базова підтримка з можливістю фільтрації та індексації)
Безпека даних	Вимагає додаткової конфігурації	Розширені можливості шифрування та аудиту	Надійна безпека, розвинені механізми контролю доступу
Резервне копіювання та відновлення	Потребує налаштування зовнішніх сервісів	Вбудована підтримка реплікації та резервного копіювання	Добре підтримується, є перевірені інструменти для бекапу і реплікації

Виходячи з наявних переваг та недоліків обраних СУБД, MariaDB була обрана для нашого інтернет-магазину ювелірних прикрас «JEWELRY LOUNGE». Вона має відмінне поєднання стабільності, продуктивності та надійності для обробки транзакційних даних у реляційних структурах. MariaDB підтримує SQL-запити, що зручно для розробки та інтеграції в Laravel, а також має високу

продуктивність для операцій CRUD [30], що важливо для великого обсягу замовлень та товарів у нашому інтернет-магазині. Крім того, вона забезпечує гарну масштабованість і безпеку, а також є безкоштовною, що робить її ідеальним вибором для проєкту.

Висновки до першого розділу

У першому розділі було визначено основні технологічні підходи та архітектурні рішення для розробки інтернет-магазину ювелірних прикрас «JEWELRY LOUNGE». Здійснено аналіз аналогічних платформ для продажу ювелірних прикрас, таких як «Золотий Вік», AURUM та Brilliant Earth, що дозволило виявити їхні переваги та недоліки. Це дозволило сформулювати вимоги до платформи, зокрема щодо інтерактивних функцій, персоналізації та зручності для користувачів.

На основі цього аналізу було обрано оптимальну архітектуру для проєкту, яка ґрунтується на клієнт-серверній моделі, де серверна частина відповідає за обробку замовлень, управління користувачами та інтеграцію з платіжними системами, а клієнтська частина взаємодіє із сервером через REST API. Також обрано MariaDB як СУБД, що забезпечує високу продуктивність, надійність і масштабованість для збереження даних про товари та замовлення.

Також розглянуто основні інструментальні засоби та програмні технології для реалізації системи, зокрема PHP та фреймворк Laravel для серверної частини. Це дозволить створити ефективну платформу для продажу ювелірних прикрас, забезпечуючи її високою продуктивністю, масштабованістю та гнучкістю для подальшого розвитку.

					ІПЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ЮВЕЛІРНИХ ПРИКРАС

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

У цьому розділі надано детальний огляд варіантів використання системи, а також описано об'єктно-орієнтовану структуру для інтернет-магазину "JEWELRY LOUNGE". Проєктована система повинна відповідати функціональним і нефункціональним вимогам, серед яких важливими є зручність користування, ефективне управління даними та можливість інтерактивного відображення результатів, зрозумілих для кінцевих користувачів.

Система поділяється на кілька основних логічних модулів, кожен з яких має свою специфічну роль. Клієнтський модуль надає користувачам інтерфейс для перегляду товарів, додавання їх до кошика, оформлення замовлень та здійснення оплати. Адміністративний модуль дозволяє адміністраторам і модераторам управляти товарами, категоріями, замовленнями, а також переглядати статистичні дані та звіти. Модуль управління правами доступу дозволяє адміністраторам розмежовувати функції для різних категорій користувачів, забезпечуючи належний рівень доступу до інформації та функціональності системи.

У системі існують три основні ролі користувачів: адміністратор, модератор і користувач. Адміністратор має повний доступ до всіх функцій системи, включаючи управління товарами, категоріями, замовленнями та перегляд аналітики. Модератор може редагувати інформацію про товари, обробляти замовлення та допомагати користувачам з їх запитамі. Користувачі можуть переглядати товар, додавати його до кошика, оформлювати замовлення та здійснювати оплату.

Бізнес-вимоги:

- Метою проєкту є створення онлайн-платформи для продажу ювелірних прикрас з можливістю детального перегляду товарів, здійснення покупок та оперативного виконання замовлень.

					ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

2. Система повинна забезпечити швидкий доступ до функцій управління товарами, замовленнями та користувачами, а також підтримувати інтуїтивно зрозумілий інтерфейс для клієнтів.

Вимоги зовнішніх користувачів (Звичайний користувач):

Користувачі мають доступ до каталогу товарів, можливість перегляду деталей товару, додавання товарів до кошика, оформлення замовлень, оплати та перегляду статусу замовлень.

Вимоги зовнішніх користувачів (Модератор):

Модератор має можливість керувати товарами, переглядати замовлення, а також забезпечувати підтримку клієнтів. Модератор має доступ до статистичних даних щодо продажів і може допомагати в управлінні категоріями товарів.

Характеристика об'єкта комп'ютеризації

Інтернет-магазин "JEWELRY LOUNGE" для продажу ювелірних прикрас, який має велику кількість користувачів, включаючи клієнтів, модераторів та адміністраторів. Система повинна забезпечувати надійне управління товарами, замовленнями, кошиками та користувачами, а також інтерактивний інтерфейс для користувачів. Веб-платформа повинна мати можливість адаптації до різних пристроїв, забезпечуючи комфортний доступ до основних функцій як на десктопах, так і на мобільних пристроях. Використання технології Blade дозволяє створити зручні та гнучкі шаблони для відображення сторінок, що спрощує інтеграцію з бекендом на Laravel. Користувачі різних ролей повинні мати доступ лише до дозволених функцій і даних, що забезпечує високий рівень безпеки та контролю доступу.

Функціональні вимоги

Система повинна дозволяти створювати, редагувати та видаляти товари, категорії, матеріали й розміри продуктів, а також обробляти замовлення та генерувати відповідні звіти для адміністратора. Крім того, система має підтримувати механізми для управління ролями користувачів і забезпечувати належний доступ до функцій системи (табл. 2.1).

					ІПЗ.КР.Б – 121 – 25 - ІПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

Функціональні вимоги до системи

Етап	Опис
Рольовий доступ (студент, медик)	Система дозволяє розмежувати функціонал між студентами та медичними працівниками відповідно до їх ролей. Студент має доступ до персональних даних, а медичний працівник - до даних усіх студентів для аналізу та управлінських рішень.
Зберігання та відображення медичних показників	Дані про стан здоров'я студентів зберігаються у базі даних та відображаються в інтерфейсі для подальшого перегляду і аналізу.
Візуалізація даних у вигляді графіків	Для швидкого аналізу даних користувачі можуть переглядати графіки змін медичних показників в динаміці, що дозволяє виявляти відхилення та оцінювати стан здоров'я.
Механізм автоматичного позначення критичних значень	Система автоматично позначає критичні медичні значення та попереджає користувачів про необхідність звернення до медичного пункту.
Прогнозування епідеміологічного ризику на основі історичних даних	Використання алгоритмів машинного навчання для прогнозування потенційних спалахів захворювань на основі історії медичних записів студентів.

Це дозволить користувачам швидко зорієнтуватися в основних функціональних вимогах системи. Таблиця також може бути зручнішою для розуміння, оскільки вона чітко розділяє кожен функцію та її опис.

Робота системи починається з реалізації рольового доступу, що дозволяє розмежувати функціонал між адміністраторами, модераторами та звичайними користувачами відповідно до їхніх повноважень [18]. Після авторизації користувачі взаємодіють з інтерфейсом, що дозволяє переглядати товари, додавати їх у кошик, оформляти замовлення та здійснювати оплату. Усі зібрані дані про користувачів та

замовлення зберігаються в базі даних і відображаються у системі у зручному вигляді.

Наступним етапом є перегляд і редагування товарів та категорій продуктів адміністраторами та модераторами. Адміністратори мають доступ до повної інформації про товари, категорії та замовлення, тоді як модератори можуть здійснювати лише перегляд та редагування інформації, не маючи доступу до адміністративних налаштувань.

Для користувачів система забезпечує можливість фільтрації товарів за різними параметрами (категорії, розміри, матеріали) та здійснення пошуку за назвою або описом товару. Крім того, система дозволяє відображати детальну інформацію про кожен товар, а також історію замовлень та статус кожного з них.

Фінальним елементом функціонального циклу є система збереження та перегляду історії замовлень, яка дозволяє користувачам стежити за своїми покупками, а адміністраторам здійснювати управління статусами замовлень та взаємодіяти з клієнтами у разі необхідності.

Таким чином, усі функціональні можливості системи тісно пов'язані між собою, утворюючи замкнутий цикл, спрямований на автоматизацію процесу покупки ювелірних прикрас та підтримку управлінських рішень в інтернет-магазині.

Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи, які впливають на зручність її використання, надійність, стабільність у роботі та здатність до масштабування. Вони не пов'язані безпосередньо з реалізацією функціоналу, але мають важливе значення для забезпечення цілісності та ефективності програмного забезпечення в реальних умовах експлуатації [19].

Система повинна бути інтуїтивно зрозумілою для різних категорій користувачів, таких як адміністратори, модератори та покупці. Інтерфейс повинен дозволяти користувачам швидко знайти необхідну інформацію та взаємодіяти з платформою без потреби в додатковому навчанні. Важливо, щоб час освоєння

					ІПЗ.КР.Б – 121 – 25 - ІПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

базового функціоналу не перевищував 15 хвилин для покупців та 30 хвилин для модераторів.

Щодо надійності, система повинна забезпечувати безвідмовну роботу протягом мінімум 30 діб. Кількість помилок у процесі використання має бути мінімальною, при цьому система повинна бути стійкою до високих навантажень, і допустимою є лише одна помилка на 1000 запитів. Це забезпечить стабільну роботу навіть при активному використанні з боку великої кількості користувачів.

Продуктивність системи повинна бути достатньою для обробки одночасних запитів від мінімум 100 активних користувачів. Це є важливою вимогою для інтернет-магазину, де може бути велика кількість одночасних покупок, переглядів товарів та оформлень замовлень.

Архітектура системи повинна забезпечувати масштабованість, що дозволяє ефективно працювати при зростанні кількості користувачів і товарів без втрати продуктивності. Масштабованість також повинна забезпечити стабільну роботу системи при збільшенні обсягу даних і підвищеному навантаженні. Таким чином, нефункціональні характеристики створюють основу для стабільної, зручної у використанні та гнучкої системи, здатної ефективно працювати в умовах реального бізнес-середовища.

Системні вимоги

Для забезпечення стабільної та повнофункціональної роботи програмного забезпечення в умовах інтернет-магазину ювелірних прикрас були визначені системні вимоги, які охоплюють як середовище виконання на стороні користувача, так і інфраструктуру серверної частини, включаючи базу даних і аналітичні інструменти. Ці вимоги є критично важливими для коректного розгортання і подальшого супроводження системи, зокрема в умовах обробки великого обсягу даних, характерного для онлайн-комерції.

Середовище виконання передбачає використання сучасних веб браузерів, таких як Chrome, Firefox або Edge, які підтримують останні версії JavaScript і HTML стандартів. Програмне забезпечення має бути сумісним із операційною системою

					ІПЗ.КР.Б – 121 – 25 - ІЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

Windows 10 або новішою. Мінімальна конфігурація клієнтського пристрою включає 2 ГБ оперативної пам'яті, процесор з тактовою частотою не менше 1,5 ГГц і щонайменше 500 МБ вільного місця на диску. Ці параметри забезпечують коректне функціонування користувацького інтерфейсу, навіть за умов обмежених апаратних ресурсів.

Щодо серверної частини, вона реалізована на основі фреймворку Laravel, який дозволяє побудувати ефективну серверну логіку та реалізувати REST API для взаємодії з клієнтською частиною. Для зберігання даних використовуються бази даних MariaDB, що забезпечують швидке збереження і обробку даних. Також передбачена можливість масштабування інфраструктури для підтримки більшої кількості користувачів, одночасних запитів і товарів у магазині. Система має інтеграцію з платіжними системами для обробки транзакцій та підтримки безпечних покупок.

Таким чином, дотримання зазначених технічних вимог дозволяє забезпечити стабільну роботу магазину, ефективний збір і обробку даних, а також гнучкість і масштабованість системи для подальшого розвитку.

Варіанти використання

Аналіз вимог користувачів дав підстави для визначення варіантів використання системи інтернет-магазину ювелірних прикрас. У нашій системі визначено три основні ролі користувачів: Користувач, Модератор і Адмін, кожна з яких має різний рівень доступу до функцій системи.

Нижче зображено побудовану діаграму варіантів використання (рис. 2.1). Діаграма показує основні функції, доступні кожному з користувачів, включаючи перегляд товарів, оформлення замовлення, управління продуктами та користувачами.

Користувач може переглядати продукти, оформляти замовлення та здійснювати покупки. Його основна взаємодія з системою - це перегляд товарів і додавання їх у кошик, а також процес оформлення замовлення.

					ІПЗ.КР.Б – 121 – 25 - ІЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

Модератор має додаткові можливості, які дозволяють йому переглядати замовлення та управляти продуктами. Це дає змогу модератору підтримувати актуальність товарів на сайті та слідкувати за виконанням замовлень.

Адміністратор має найвищий рівень доступу, включаючи управління користувачами, перегляд замовлень та адміністрування товарів. Адміністратор забезпечує повний контроль над системою.

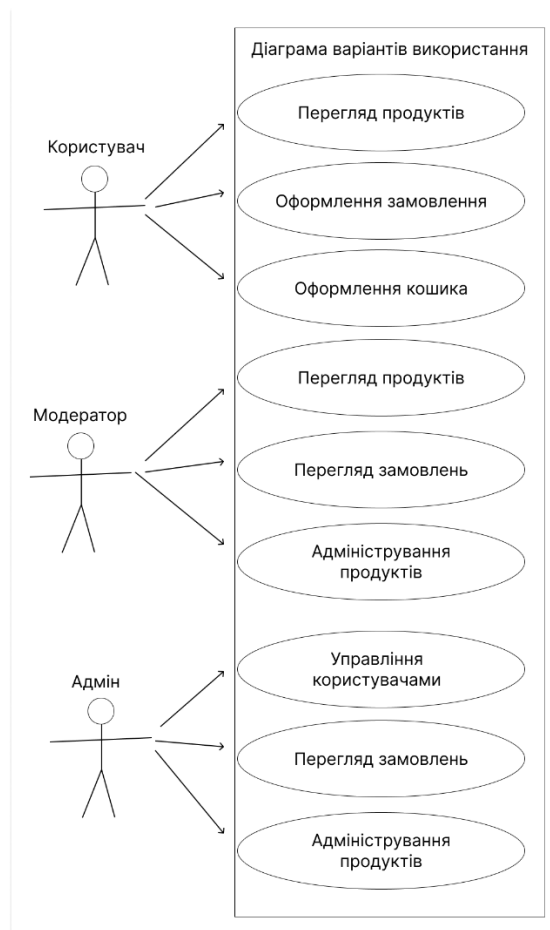


Рис. 2.1 - Діаграма варіантів використання

Опис діаграми варіантів використання наведено в таблицях (табл. 2.2 - 2.7). Кожна таблиця детально ілюструє конкретний сценарій взаємодії користувачів із системою відповідно до їх ролей. Це дозволяє чітко визначити функціональні можливості, доступні для кожного типу користувача, а також межі доступу до інформації.

Таблиці також допомагають з'ясувати, як різні ролі взаємодіють із ключовими функціями платформи, такими як управління продуктами, обробка замовлень і управління користувачами. Цей підхід дає змогу організувати

структуру інтерфейсу та навігації відповідно до потреб кожного користувача, що полегшує інтеграцію та підтримку системи в майбутньому.

Таблиця 2.2

Опис преценденту «Перегляд продуктів»

Назва варіанту використання	Перегляд продуктів
Основні актори	Користувач, Модератор, Адмін
Короткий опис	Користувач може переглядати доступні ювелірні вироби, їхні характеристики, ціни та зображення. Модератори та адміністратори можуть також редагувати, додавати чи видаляти продукти. Всі користувачі можуть використовувати фільтри для швидкого пошуку товарів.

Цей сценарій дозволяє користувачам зручно переглядати продукцію в магазині та отримувати актуальну інформацію про ювелірні вироби. Користувачі можуть сортувати товари за різними критеріями, такими як ціна, матеріал або популярність, що полегшує процес пошуку. Модератори та адміністратори, крім того, можуть редагувати або додавати нові продукти, підтримуючи актуальність асортименту. Всі зміни в продуктах автоматично оновлюються на сайті, забезпечуючи користувачів актуальними даними для покупки.

Таблиця 2.3

Опис преценденту «Оформлення замовлення»

Назва варіанту використання	Оформлення замовлення
Основні актори	Користувач
Короткий опис	Користувач може додавати продукти до кошика, переглядати його вміст, змінювати кількість товару та оформляти замовлення, вибираючи метод оплати та доставки. Користувач також має можливість переглядати статус своїх замовлень.

Цей сценарій дозволяє користувачам завершити покупку, вибравши необхідні товари та методи оплати. Після оформлення замовлення система генерує підтвердження та інформує користувача про успішне завершення процесу. Крім

					ІПЗ.КР.Б – 121 – 25 - ІЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

того, користувач отримує можливість відстежувати статус свого замовлення, що забезпечує додаткову зручність при покупках. Весь процес оформлення замовлення інтегрований з платіжною системою, що гарантує безпеку і швидкість транзакцій.

Таблиця 2.4

Опис преценденту «Адміністрування продуктів»

Назва варіанту використання	Адміністрування продуктів
Основні актори	Модератор, Адмін
Короткий опис	Адмін та модератор мають можливість додавати нові продукти, редагувати або видаляти існуючі товари, змінювати ціни, додавати нові категорії та матеріали, а також управляти їх характеристиками.

Цей сценарій дає адміну та модератору повний контроль над асортиментом товарів у магазині. Вони можуть оперативно оновлювати інформацію про продукти, додаючи нові моделі або видаляючи застарілі. Крім того, адміністратори можуть налаштовувати доступ до різних категорій продуктів, що дозволяє ефективно організувати товарний каталог. Такий контроль над контентом забезпечує постійну актуальність і відповідність асортименту потребам користувачів. Це дозволяє забезпечити гнучкість у управлінні продуктами та швидку адаптацію до змін у попиті та трендах на ринку.

Таблиця 2.5

Опис преценденту «Перегляд замовлень»

Назва варіанту використання	Перегляд замовлень
Основні актори	Модератор, Адмін
Короткий опис	Адмін та модератор можуть переглядати всі замовлення, перевіряти статус кожного замовлення, оновлювати статуси та здійснювати інші операції з замовленнями (наприклад, відміну або обробку).

Цей сценарій дозволяє модераторам та адміністраторам управляти замовленнями, підтримуючи належний рівень обслуговування клієнтів.

Таблиця 2.6

Опис преценденту «Управління користувачами»

Назва варіанту використання	Управління користувачами
Основні актори	Адмін
Короткий опис	Адмін має можливість переглядати, створювати, редагувати або видаляти користувачів, призначати їм відповідні ролі (адміністратор, модератор, користувач) та управляти доступом до різних частин магазину в залежності від їхніх ролей.

Цей сценарій дає адміністратору контроль над правами доступу всіх користувачів магазину.

Таблиця 2.7

Опис преценденту «Оформлення кошика»

Назва варіанту використання	Отримання сповіщень про критичні випадки
Основні актори	Користувач
Короткий опис	Користувач може переглядати товари, що знаходяться в його кошику, змінювати кількість, видаляти товари та продовжувати оформлення замовлення, включаючи вибір способу оплати та доставки.

Цей сценарій дозволяє користувачу безперешкодно завершити покупки після того, як всі товари додані до кошика.

Щоб ефективно реалізувати функціональні можливості інтернет-магазину, необхідно організувати чітку внутрішню структуру системи, яка буде відповідати сценаріям використання, забезпечуючи стабільність і ефективність роботи. Це включає в себе розділення логіки в межах серверної частини застосунку, де кожен компонент або підсистема відповідатиме за певний набір функціональностей. Для цього використовуються різні моделі, сервіси та контролери, що забезпечують ефективну взаємодію між базою даних і користувачем.

Умовні "класи" системи (тобто логічно відокремлені компоненти) визначені відповідно до їх функціонального призначення. До них належать як моделі

продуктів, замовлень, користувачів, так і сервіси, відповідальні за аналітику, управління кошиками, обробку замовлень, управління ролями користувачів тощо. Це дозволяє забезпечити розширюваність і підтримуваність програмного коду, а також дозволяє ефективно додавати нові функціональності без порушення цілісності системи.

Основні структурні елементи системи, які відповідають за реалізацію ключових задач в інтернет-магазині (табл. 2.8). Кожен елемент виконує свою специфічну роль у забезпеченні стабільної роботи магазину, оптимізації взаємодії користувачів із продуктами та замовленнями, а також управлінні правами доступу до різних частин системи.

Таблиця 2.8

Опис основних класів і компонентів системи

Клас	Опис
Cart	Відповідає за кошик користувача, зберігає продукти, які користувач хоче придбати, та зв'язок з конкретним користувачем.
Order	Моделює замовлення користувача, зберігає інформацію про продукти, їх кількість, статуси, адресу доставки, телефон та інші дані.
Product	Модель продукту, що містить всі необхідні дані про товар, такі як ціна, опис, категорія, матеріал та зображення.
ProductCategory	Категорії продуктів для зручності фільтрації товарів. Містить зв'язок із продуктами, що належать до цієї категорії.
ProductMaterial	Моделює матеріали, з яких виготовлені продукти (наприклад, золото, срібло, діамант).
User	Модель користувача, що відповідає за управління акаунтами користувачів, їх ролями та правами доступу.

ProductSize	Розміри продуктів, що дозволяють сортувати та фільтрувати товари за відповідними розмірами, особливо для товарів типу "ювелірні вироби".
Role	Визначає ролі користувачів (адміністратор, модератор, звичайний користувач), кожен із яких має відповідні права на виконання операцій.
Permission	Описує конкретні дозволи для кожної ролі користувача, такі як перегляд, додавання, оновлення та видалення продуктів, замовлень, користувачів.
OrderController	Керує обробкою замовлень, включаючи створення, перегляд, оновлення та видалення замовлень користувачів.
ProductController	Керує операціями з продуктами, включаючи додавання, редагування, перегляд і видалення товарів.
CartController	Керує кошиками покупок, дозволяючи додавати, змінювати кількість товарів або видаляти їх з кошика.

Ці класи та компоненти допомагають забезпечити ефективне та стабільне функціонування інтернет-магазину, кожен з них виконує свою спеціалізовану роль, зберігаючи чітке розмежування обов'язків завдяки чіткому розмежуванню дозволів між ролями [9] і забезпечуючи зручність для розширення та модифікації системи в майбутньому.

2.2. Розробка бази даних системи

Для зберігання даних в інтернет-магазині ювелірних прикрас була розроблена реляційна база даних, що використовує MySQL. Ця СУБД забезпечує ефективне збереження даних про продукти, замовлення, користувачів і їхню

взаємодію з магазином, а також дозволяє здійснювати складні запити для аналітики і відображення результатів на сайті.

База даних побудована з урахуванням принципів нормалізації, чіткого розподілу відповідальності між таблицями та забезпечення масштабованості для майбутнього розширення. Структура БД охоплює ключові аспекти роботи інтернет-магазину: облік користувачів, управління продуктами, категоріями, замовленнями та кошиками покупок.

Далі представлено таблицю (табл. 2.9) яка узагальнює опис структури бази даних інтернет-магазину.

Таблиця 2.9

Структура бази даних

Назва таблиці	Призначення
users	Зберігає інформацію про користувачів (ім'я, email, роль, пароль).
products	Описує продукти в магазині (назва, ціна, опис, категорія, матеріали, кількість на складі).
orders	Містить дані про замовлення користувачів (статус замовлення, адреса доставки, телефон, продукти в замовленні).
order_has_products	Таблиця для зв'язку замовлень з продуктами (кількість продуктів в замовленні).
cart	Зберігає дані про кошики користувачів (користувач, продукти в кошику).
cart_has_product	Зв'язує кошик з продуктами (продукт, кількість в кошику).
cart_has_product	Категорії продуктів (назва, опис).
product_sizes	Розміри продуктів (назва, категорія).
product_materials	Матеріали продуктів (назва, опис, колір).
product_has_materials	Зв'язок між продуктами і матеріалами (продукт, матеріал).

Кожна таблиця спроектована таким чином, щоб забезпечити логічну цілісність даних, дозволяючи проводити фільтрацію та аналіз даних, такі як пошук по категоріях, матеріалах, замовленнях чи користувачах. Всі зв'язки між таблицями підтримуються через зовнішні ключі, що забезпечує узгодженість даних у різних таблицях та полегшує взаємодію між даними.

Розпочнемо опис з таблиці users (табл. 2.10). Вона зберігає інформацію про користувачів інтернет-магазину, таких як ім'я, електронна пошта, пароль і роль у системі. Вона забезпечує облік користувачів і їх доступ до різних частин магазину в залежності від їх ролі (адмін, модератор, користувач).

Таблиця 2.10

Опис таблиці users

Поле	Тип	Призначення
id	int	Ідентифікатор користувача
name	string	Ім'я користувача
email	string	Електронна пошта користувача
password	string	Пароль користувача
remember_token	string	Токен для запам'ятовування користувача
created_at	timestamp	Дата створення запису
updated_at	timestamp	Дата останнього оновлення запису

Таблиця products є основною для зберігання інформації про кожен товар в інтернет-магазині, включаючи не тільки загальні характеристики, такі як ціна, опис та зображення, але й важливі дані щодо доступності продукції на складі. Всі товари, представлені на платформі, зберігаються в цій таблиці і асоціюються з іншими сутностями, такими як категорії, матеріали та розміри, через зовнішні ключі. Це дозволяє зручно організувати фільтрацію та пошук продуктів, забезпечуючи користувачам легкий доступ до необхідних товарів за допомогою різних фільтрів та категорій. Важливість цієї таблиці полягає в тому, що вона забезпечує зберігання основних даних про продукти та їх взаємозв'язки з іншими елементами системи, такими як категорії товарів, матеріали та їхні характеристики (табл. 2.11).

Таблиця також містить дані про кількість продукту на складі, що дозволяє автоматично контролювати рівень запасів і відображати актуальну інформацію про наявність товару. Це забезпечує зручність як для адміністраторів, які можуть

оперативно оновлювати кількість товарів, так і для користувачів, які отримують точну інформацію про доступність продукції під час здійснення покупок.

Таблиця 2.11

Опис таблиці products

Поле	Тип	Призначення
id	bigincrements	Унікальний ідентифікатор продукту
name	string	Назва продукту
description	text	Опис продукту
price	decimal	Ціна продукту
preview	string	Посилання на основне зображення продукту
stock_quantity	integer	Кількість продукту на складі
product_category_id	unsignedBigInteger	Зовнішній ключ, який зв'язує продукт з його категорією
created_at	timestamp	Дата створення продукту
updated_at	timestamp	Дата останнього оновлення продукту

Таблиця orders відображає всі аспекти обробки замовлення, починаючи від даних користувача, який здійснив покупку, і до статусу самого замовлення. Вона дозволяє відслідковувати історію покупок кожного користувача, а також контролювати процес виконання замовлень на різних етапах - від оформлення до доставки.

Кожне замовлення містить детальну інформацію про продукт, що включає не лише адресу доставки, але й можливість додавання спеціальних приміток, що можуть бути враховані під час обробки. Завдяки зв'язку із користувачем, система дозволяє швидко визначити, які товари були придбані та які статуси були призначені для замовлення.

Ця таблиця також служить основою для виведення звітів, що дозволяють адміністраторам та модераторам ефективно відслідковувати прогрес замовлень, а

також швидко реагувати на зміни статусу або виявлені проблеми з доставкою (табл. 2.12).

Таблиця 2.12

Опис таблиці orders

Поле	Тип	Призначення
id	bigincrements	Ідентифікатор замовлення
user_id	unsignedBigInteger	Зовнішній ключ, що зв'язує замовлення з користувачем
status	enum	Статус замовлення (наприклад, "оплачено", "обробляється", "відправлено")
payment_status	decimal	Статус платежу
total_price	decimal	Загальна сума замовлення
shipping_address	string	Адреса доставки
shipping_city	string	Місто доставки
shipping_postal_code	string	Поштовий код доставки
shipping_country	string	Країна доставки
phone	string	Телефон для контакту
notes	text	Додаткові примітки до замовлення
placed_at	datetime	Дата оформлення замовлення
created_at	timestamp	Дата створення замовлення
updated_at	timestamp	Дата останнього оновлення замовлення

Таблиця product_category дозволяє ефективно організувати товари за категоріями, що спрощує процес навігації для користувачів та дозволяє здійснювати швидкий пошук серед ювелірних прикрас. Кожна категорія має свою унікальну назву, опис та зображення, що сприяє кращому представлення товарів у каталозі. Завдяки використанню поля slug, категорії можна зручно ідентифікувати в URL, що також покращує SEO оптимізацію сайту (табл. 2.13).

Ця таблиця також дозволяє ефективно управляти асоціацією товарів з відповідними категоріями, що спрощує процес адміністративного управління та оновлення категорій у системі.

Таблиця 2.13

Опис таблиці product_category

Поле	Тип	Призначення
id	bigincrements	Унікальний ідентифікатор категорії
name	string	Назва категорії
slug	string	Слово або фраза, яка використовується в URL для категорії
description	text	Опис категорії
image	string	Зображення категорії
created_at	timestamp	Дата створення категорії
updated_at	timestamp	Дата останнього оновлення категорії

Таблиця product_material зберігає інформацію про матеріали ювелірних прикрас, такі як золото, срібло, діаманти тощо (табл. 2.14).

Таблиця 2.14

Опис таблиці product_material

Поле	Тип	Призначення
id	bigincrements	Унікальний ідентифікатор матеріалу
name	string	Назва матеріалу
description	string	Опис матеріалу
color	string	Колір матеріалу
created_at	timestamp	Дата створення матеріалу
updated_at	timestamp	Дата останнього оновлення матеріалу

Таблиця `order_has_products` зв'язує продукти із замовленнями. Відображає, які продукти були придбані в кожному замовленні та в якій кількості (табл. 2.15).

Таблиця 2.15

Опис таблиці `order_has_products`

Поле	Тип	Призначення
<code>product_id</code>	<code>unsignedBigInteger</code>	Зовнішній ключ, що посилається на продукт
<code>order_id</code>	<code>unsignedBigInteger</code>	Зовнішній ключ, що посилається на замовлення
<code>quantity</code>	<code>integer</code>	Кількість продукту в замовленні
<code>price</code>	<code>decimal</code>	Ціна продукту на момент оформлення замовлення
<code>created_at</code>	<code>timestamp</code>	Дата створення матеріалу
<code>updated_at</code>	<code>timestamp</code>	Дата останнього оновлення матеріалу

Діаграма бази даних (рис. 2.2) ілюструє зв'язки між основними таблицями, які утворюють структуру даних для інтернет-магазину ювелірних прикрас. Вона відображає, як користувачі, продукти, замовлення, категорії, матеріали та інші компоненти взаємодіють між собою, що забезпечує коректне функціонування системи. Зокрема, таблиця `"orders"` зберігає дані про замовлення, пов'язані з конкретними користувачами, в той час як `"product_has_materials"` показує зв'язок між продуктами та матеріалами. Це дозволяє ефективно управляти даними, що необхідні для підтримки магазинного процесу, від вибору товару до завершення транзакції. Така структура також забезпечує масштабованість і гнучкість, дозволяючи легко додавати нові функціональні можливості в майбутньому.

Діаграма також показує, як різні таблиці, такі як `"cart_has_product"` і `"order_has_products"`, забезпечують зв'язок між кошиком користувача і замовленими товарами, що дозволяє зберігати інформацію про кожен товар, доданий до замовлення. Така організація даних дозволяє забезпечити точне

відстеження кожного етапу покупки і зберігати всі необхідні деталі для ефективного виконання замовлення та управління асортиментом.



Рис. 2.2 - База даних системи

Таке проєктування бази даних забезпечує ефективний доступ до інформації, сумісність із ORM-інструментами, можливість масштабування системи для обробки зростаючого обсягу даних та підвищення ефективності аналітичних запитів. База даних дозволяє зберігати дані про користувачів, продукти, замовлення, категорії та матеріали, що забезпечує зручну роботу інтернет-магазину та його адміністрування.

2.3. Проектування та реалізація алгоритмів роботи системи

Система інтернет-магазину ювелірних прикрас реалізує алгоритми і механізми, які забезпечують ефективну обробку даних про товари, замовлення, кошики, а також оплату через інтеграцію з платіжними системами. Завдяки цим механізмам користувачі можуть здійснювати покупки, модератори та адміністратори мають можливість управляти товарами, а система автоматично обробляє замовлення та виводить дані на фронтенд для користувачів.

Після авторизації на сайті користувач потрапляє на свою домашню сторінку. В залежності від ролі, визначеної в базі даних, йому надається доступ до різних функцій. Для студентів вхід здійснюється через email, а для адміністраторів та модераторів - через логін і пароль.

Інтерфейс системи має навігаційну панель, де користувач може перейти до розділів профілю, кошика, продуктів, замовлень і категорій. Для адаптації інтерфейсу під мобільні пристрої, є кнопка згортання панелі, що спрощує перегляд на маленьких екранах.

Далі опишемо наші алгоритми. Розпочнемо з алгоритму перегляду та управління кошиком.

Користувач може додавати продукти до кошика, переглядати їх і змінювати кількість. Якщо кошик порожній, система виводить повідомлення, що кошик потребує заповнення для оформлення замовлення. Після додавання товарів у кошик, користувач має можливість переходити до оформлення замовлення. Важливим моментом є обчислення загальної вартості замовлення, яка базується на сумі вартості продуктів та кількості кожного товару (лістинг 2.1).

```
public function addToCart(Request $request, Product $product): RedirectResponse
{
    $cart = Cart::firstOrCreate(['user_id' => auth()->id()]);
    $cartProduct = $cart->products()->where('product_id', $product->id)->first();

    if ($cartProduct) {
        $cart->products()->updateExistingPivot($product->id, [
            'quantity' => $cartProduct->pivot->quantity + 1
        ]);
    } else {
        $cart->products()->attach($product->id, ['quantity' => 1]);
    }
}
```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    return redirect()->route('carts.index')->with('success', 'Товар додано до кошику!');
}

```

Лістинг 2.1 - Перегляд та управління кошиком

Цей фрагмент коду відповідає за додавання продукту до кошика. Якщо продукт вже є в кошику, кількість оновлюється, якщо немає - товар додається з кількістю 1.

Після перевірки кошика система дозволяє користувачеві перейти до оформлення замовлення. Алгоритм перевіряє, чи є продукти в кошику. Якщо вони є, обчислюється загальна сума на основі кількості та ціни кожного товару в кошику. Потім користувач вводить необхідні дані для доставки, такі як адреса, місто, поштовий код, країна та телефон. Крім того, система надає можливість вибору методу доставки та оплати. Після цього алгоритм автоматично генерує нове замовлення, зберігає його в базі даних і надає користувачеві підтвердження про успішне створення замовлення. Далі користувач перенаправляється на сторінку оплати, де він може вибрати зручний спосіб завершення транзакції (лістинг 2.2).

```

public function store(OrderStoreRequest $request): RedirectResponse
{
    $validated = $request->validated();

    $order = Order::create([
        'user_id' => auth()->id(),
        'status' => 'pending',
        'payment_status' => 'unpaid',
        'total_price' => Cart::where('user_id', auth()->id())->first()->products->sum(function ($product) {
            return $product->price * $product->pivot->quantity;
        })),
        'shipping_address' => $request->shipping_address,
        'shipping_city' => $request->shipping_city,
        'shipping_postal_code' => $request->shipping_postal_code,
        'shipping_country' => $request->shipping_country,
        'phone' => $request->phone,
        'notes' => $request->notes,
        'placed_at' => now(),
    ]);

    $cart = Cart::with('products')->where('user_id', auth()->id())->first();

    foreach ($cart->products as $product) {
        $order->products()->attach($product->id, [
            'quantity' => $product->pivot->quantity,
            'price' => $product->price,
        ]);
    }
}

```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    return redirect()->route('checkout.payment', $order)->with('success', 'Замовлення створено!');
}

```

Лістинг 2.2 - Оформлення замовлення

Після виконання цієї операції, замовлення буде збережено в базі даних, і користувач отримає підтвердження про успішне оформлення замовлення. Однак перед тим, як відправити його на платіжну сторінку, система перевіряє правильність заповнення даних і забезпечує правильний запис у базі.

Алгоритм оплати через платіжну систему LiqPay реалізується через окремий сервіс, який генерує форму для оплати і обробляє callback-дані після завершення транзакції. Після успішної оплати статус замовлення оновлюється на "paid", а продукти видаляються з кошика (лістинг 2.3).

```

public function payment(Order $order, LiqPayService $liqPayService)
{
    $form = $liqPayService->getForm([
        'action'    => 'pay',
        'amount'    => $order->total_price,
        'currency'  => 'UAH',
        'description' => 'Замовлення №' . $order->id,
        'order_id'  => $order->id,
        'version'   => 3,
        'language'  => 'uk',
        'result_url' => route('checkout.success', $order),
        'server_url' => route('checkout.callback', $order),
    ]);

    return view('front.pages.payment', compact('form', 'order'));
}

```

Лістинг 2.3 - Формування форми для оплати

Описаний алгоритм оплати через платіжну систему LiqPay реалізується за допомогою окремого сервісного класу, який відповідає за генерацію HTML-форми для ініціації транзакції та обробку відповіді після її завершення. Метод payment() приймає об'єкт замовлення та сервіс LiqPay, після чого викликає метод getForm(), що створює форму з параметрами, необхідними для оплати. Серед них – сума замовлення, яка береться з властивості total_price, валюта платежу (у цьому випадку гривня), опис із номером замовлення, версія API, мова інтерфейсу форми, а також два посилання: result_url, на який буде перенаправлено користувача після

					ПЗ.КР.Б – 121 – 25 - ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

завершення оплати, та `server_url`, який LiqPay використовує для відправлення серверної відповіді (callback) із підтвердженням транзакції.

Сформована HTML-форма передається до представлення `front.pages.payment`, де вона буде виведена на сторінку, що дозволяє користувачу завершити оплату, не покидаючи меж сайту. Після цього, якщо оплата проходить успішно, серверна частина сайту через callback-метод обробляє отримані від LiqPay дані, змінює статус замовлення на "paid", що фіксує його як оплачене, та очищає кошик користувача від відповідних товарів. Такий підхід дозволяє не лише забезпечити безпеку та надійність оплати, а й гарантує безперервність процесу оформлення замовлення з точки зору користувацького досвіду.

Контроль доступу та матриця прав

У системі інтернет-магазину ювелірних прикрас реалізовано контроль доступу за ролями. Це дозволяє чітко розмежувати права кожного користувача в залежності від його ролі [25]. Адміністратори мають повний доступ до всіх функцій системи, модератори можуть керувати продуктами та категоріями, а користувачі мають обмежений доступ, обмежений лише до власного профілю, кошика та оформлення замовлень (табл. 2.16).

Діаграма також показує, як різні таблиці, такі як "cart_has_product" і "order_has_products", забезпечують зв'язок між кошиком користувача і замовленими товарами, що дозволяє зберігати інформацію про кожен товар, доданий до замовлення. Така організація даних дозволяє забезпечити точне відстеження кожного етапу покупки і зберігати всі необхідні деталі для ефективного виконання замовлення та управління асортиментом. У цій таблиці вказано рівні доступу для кожної ролі. Це дозволяє адмінам ефективно контролювати доступ до чутливих даних та функцій, забезпечуючи безпеку та правильне управління правами користувачів.

Значення в таблиці:

0 - відсутній доступ;

1 - доступ до власних даних;

					ІПЗ.КР.Б – 121 – 25 - ІЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

2 - повний доступ до всіх відповідних даних.

Таблиця 2.16

Матриця доступу до функцій системи

Функція	Адміністратор	Модератор	Користувач
Перегляд особистого профілю	1	1	1
Управління товарами	2	2	0
Управління категоріями продуктів	2	2	0
Перегляд списку користувачів	2	1	0
Перегляд замовлень	2	2	1
Створення та редагування продуктів	2	2	0
Оформлення замовлення	0	0	1
Перегляд кошика	0	0	1
Оплата замовлення	0	0	1

Основні функції, доступні кожному типу користувача, чітко розмежовані. Наприклад, адмін може керувати всіма аспектами сайту, включаючи користувачів і продукти, в той час як модератор має доступ до редагування та видалення товарів, а також перегляду замовлень. Користувачі мають доступ лише до оформлення замовлень і перегляду свого профілю та кошика.

Діаграма активності (рис. 2.3) наочно показує, як різні ролі користувачів взаємодіють з системою інтернет-магазину. Кожна роль має власний набір доступних функцій, що забезпечує відповідний рівень доступу до даних та можливостей для здійснення покупок, управління товарами, замовленнями та користувачами.

Ця діаграма допомагає чітко розрізняти ролі адміністратора, модератора та користувача, що сприяє ефективному управлінню правами доступу. Вона демонструє логіку переходів між етапами взаємодії з інтерфейсом, що забезпечує користувачам зручний та інтуїтивно зрозумілий досвід.



Рис. 2.3 - Діаграма активності

Висновки до другого розділу

У процесі проєктування інтернет-магазину ювелірних прикрас були визначені основні варіанти використання системи відповідно до ролей користувачів: Адмін, Модератор та Користувач. На основі цих вимог була спроектована об'єктно-орієнтована структура системи, а також визначено взаємодію між її ключовими компонентами. Особлива увага була приділена розробці реляційної бази даних, яка ефективно зберігає дані про користувачів, продукти, замовлення та кошики покупок. Структура бази даних забезпечує масштабованість, надійність та швидкість обробки запитів.

Окрім того, було розроблено алгоритми роботи системи, що включають автоматичне збирання і обробку даних про замовлення, керування товарами та категоріями, а також інтеграцію платіжних сервісів. Завдяки цьому реалізовано зручний процес для користувачів щодо перегляду товарів, додавання їх до кошика, оформлення та оплати замовлень.

Таким чином, система забезпечує чітку структуру доступу і функціоналу для різних ролей користувачів, що дозволяє ефективно управляти продуктами, замовленнями та користувачами, а також забезпечує безпечний та зручний процес покупок для кінцевих споживачів.

					ІПЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІНТЕРНЕТ-МАГАЗИНУ ЮВЕЛІРНИХ ПРИКРАС

3.1. Порядок встановлення та налаштування параметрів системи

Розроблена система інтернет-магазину ювелірних прикрас реалізована як веб-застосунок на базі фреймворку Laravel для серверної частини та Blade для відображення клієнтської частини. Встановлення та запуск системи здійснюється локально або на серверному середовищі з підтримкою PHP, MySQL.

Перед запуском необхідно встановити наступне програмне забезпечення:

- PHP (версія 8.1 або новіша);
- MySQL Server (версія 8 або новіша);
- Composer для керування PHP-залежностями [7];
- Git для клонування репозиторію [8].

Після підготовки середовища користувач має клонувати репозиторій системи з GitHub або іншого сховища коду. Структура проєкту містить кілька основних каталогів: app (серверна частина), resources/views (Blade шаблони для клієнтської частини), database (міграції та сидери для бази даних) і конфігураційні файли.

Серверна частина розроблена за допомогою фреймворку Laravel. У конфігураційному файлі .env зазначаються параметри підключення до бази даних (лістинг 3.1).

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=jewelry_db
DB_USERNAME=root
DB_PASSWORD=""
```

Лістинг 3.1 - Параметри підключення до бази даних

Після цього необхідно встановити залежності проєкту за допомогою команди composer install. Для запуску серверної частини у режимі розробки використовуємо команду php artisan serve. Сервер буде доступний за адресою http://127.0.0.1:8000.

Клієнтська частина використовує Blade шаблони для відображення веб-сторінок.

Перейдемо до налаштування бази даних. Після встановлення залежностей потрібно ініціалізувати базу даних. Структура бази даних відповідає таблицям описаним у розділі 2.4. Для створення таблиць виконується команда `php artisan migrate`. Щоб заповнити базу даних тестовими даними, використовується сидер, для цього треба запустити команду `php artisan db:seed`.

Після налаштування серверної частини, клієнтської частини та бази даних, система готова до роботи. Користувачі можуть авторизовуватися у системі, переглядати продукцію, додавати її до кошика, оформляти замовлення та здійснювати оплату.

Як альтернативу ручному налаштуванню середовища, можна використовувати Docker-контейнери [27], які дозволяють автоматизувати процес розгортання та забезпечують портативність конфігурацій.

Отже, налаштування системи включає в себе підготовку середовища, встановлення необхідних залежностей, запуск сервера, налаштування бази даних і запуск аналітичного модуля для обробки та прогнозування даних.

3.2. Структура інтерфейсу

Інтерфейс розробленої системи інтернет-магазину ювелірних прикрас спроектований з урахуванням принципів простоти та зручності використання, що забезпечує користувачам комфортну взаємодію як на десктопах, так і на мобільних пристроях. Всі елементи інтерфейсу взаємодіють між собою в єдиному веб-застосунку на основі Laravel та Blade шаблонів, що забезпечує адаптивний дизайн.

Після переходу на сайт користувач потрапляє на сторінку авторизації, де може увійти до системи за допомогою email та пароля, або через обліковий запис соціальних мереж. Для користувачів різних ролей - адміністраторів, модераторів та покупців - передбачено різний рівень доступу до функцій. Сторінка входу виглядає наступним чином (рис. 3.1).

					ІПЗ.КР.Б – 121 – 25 - ІЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

Рис. 3.1 - Сторінка входу

Після авторизації користувач потрапляє на головну сторінку, де розміщуються основні категорії товарів, акційні пропозиції, а також популярні ювелірні вироби, що допомагає зразу орієнтуватися в асортименті магазину. Ця сторінка також містить банери з сезонними знижками та новими надходженнями, що надає можливість користувачам швидко знайти бажані товари або дізнатися про актуальні пропозиції. Головна сторінка візуально приваблива, зручна для навігації, а також надає можливість зручного пошуку товарів за допомогою фільтрів або категорій, що робить процес покупок максимально зручним та ефективним.

У розділі "Каталог товарів" користувач може переглядати всю доступну продукцію, де всі вироби розподілені за категоріями, матеріалами, розмірами та іншими важливими характеристиками, що дозволяє детально ознайомитись з усіма варіантами. Каталог адаптований для зручності пошуку, даючи змогу користувачам швидко знайти потрібний товар завдяки системі фільтрації та сортування. На сторінці кожного товару представлені детальні характеристики, які включають матеріали, розміри, ціни, фотографії та опис, а також кнопка для додавання товару

					ІПЗ.КР.Б – 121 – 25 - ІЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

до кошика. Всі ці функції працюють синхронно, щоб забезпечити користувачам зручний і приємний досвід покупок (рис. 3.2).

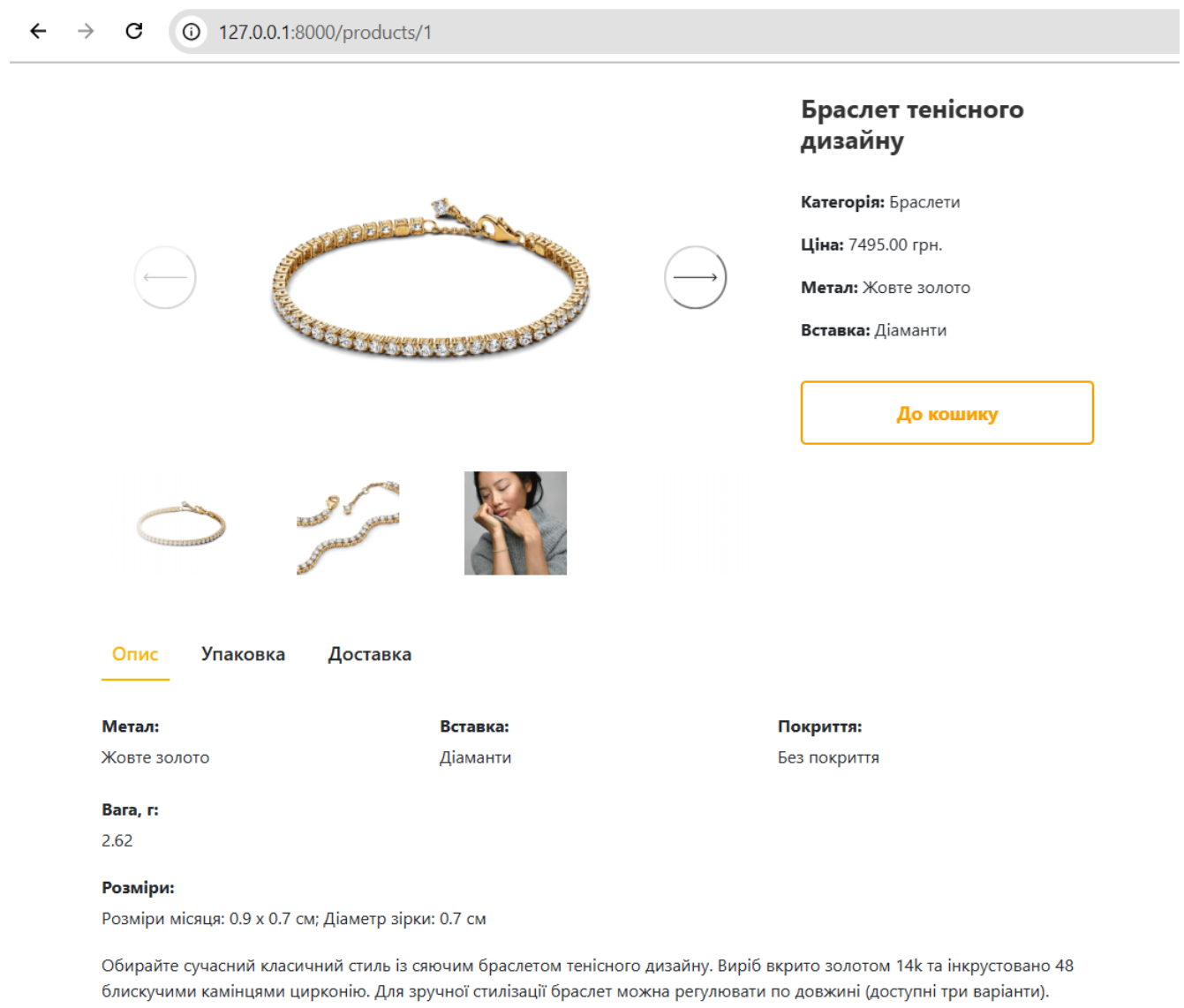


Рис. 3.2 - Сторінка товару

Після додавання товарів у кошик користувач може перейти до сторінки кошика, де відображаються всі вибрані товари, їх кількість, ціна та загальна сума. Користувач має можливість змінювати кількість товарів, видаляти їх або перейти до оформлення замовлення (рис. 3.3). Крім того, система автоматично оновлює загальну суму замовлення після кожної зміни в кошику, що дає користувачеві повний контроль над його покупками. На цій сторінці також відображаються варіанти доставки та інші додаткові послуги, які можна вибрати перед завершенням покупки.

					ІПЗ.КР.Б – 121 – 25 - ІПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

127.0.0.1:8000/cart

Jewelry Lounge

Каталог Про нас Політика приватності Admin

Кошик

Назва	Ціна	Кількість	Сума	Дія
Браслет "Блискуче паве"	6795.00 грн.	4	20385 грн.	Оновити Видалити
Браслет I-D Pandora Signature	4247.00 грн.	1	4247 грн.	Оновити Видалити

Загальна сума: 24632 грн.

Оформити замовлення

Контакти
Телефон: +38 (099) 123-45-67
Час роботи: Пн-Пт: 9:00 - 18:00, Сб-Нд: Вихідний
Оплата: Ми приймаємо оплату через ПриватБанк, Google Pay та інші способи.

Каталог

Кольє

Браслети

Підвіски та кулони

Сережки

Каблучки

Рис. 3.3 - Сторінка кошика

Перейшовши до сторінки оформлення замовлення, користувач вводить свої дані для доставки, обирає спосіб оплати та підтверджує замовлення. Після підтвердження, система автоматично відправляє користувачу дані для оплати, а також інформацію про замовлення (рис. 3.4).

127.0.0.1:8000/checkout

Оформлення замовлення

Назва товару	Ціна	Кількість	Сума
Браслет тенісного дизайну	10.00 грн.	3	30 грн.

Загальна сума: 30 грн.

Введіть свої дані для доставки

Адреса доставки

Провулок 5 Київський 33

Місто

Житомир

Поштовий індекс

10001

Країна

Ukraine

Телефон

0937581777

Примітки

відправити швидше

Оплатити зараз

Рис. 3.4 - Сторінка оформлення замовлення

					ПЗ.КР.Б – 121 – 25 - ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

Після того, як користувач переходить на сторінку оплати (рис. 3.5), він має можливість вибрати спосіб оплати через платіжні системи, такі як LiqPay. Користувач може оплатити замовлення карткою або через інші способи оплати.

Рис. 3.5 - Сторінка оплати через LiqPay

Після успішної оплати користувач отримує підтвердження через сповіщення на сайті та на електронну пошту. На сторінці показується номер замовлення, сума та можливість завантажити квитанцію (рис. 3.6).

Рис. 3.6 - Підтвердження оплати

					ІПЗ.КР.Б – 121 – 25 - ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

Інтерфейс системи підтримує адаптивну верстку, тому сторінки коректно відображаються на мобільних пристроях. Всі елементи автоматично підлаштовуються під різні розміри екранів, що забезпечує зручність використання на будь-яких пристроях (рис. 3.7).

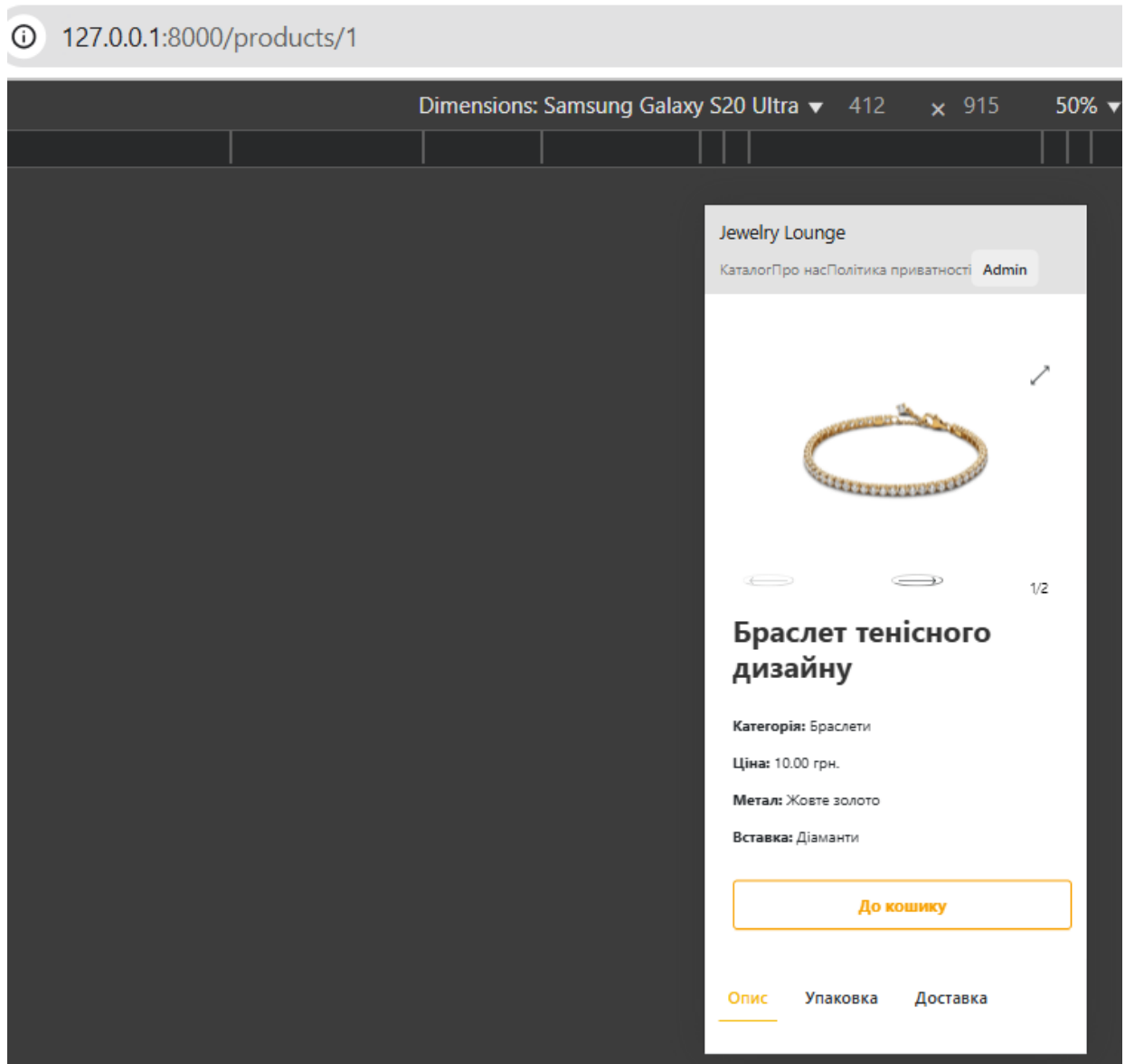


Рис. 3.7 - Адаптивний дизайн

Адмінам та модераторам надано доступ до окремої адміністративної панелі, де вони можуть здійснювати управління товарами, користувачами, замовленнями та іншими важливими даними. Сторінка для управління товарами дозволяє переглядати весь список наявних товарів, додавати нові, редагувати існуючі або видаляти їх. Кожен товар має детальну інформацію про назву, категорію, матеріал, розмір, ціну та кількість на складі.

					ІПЗ.КР.Б – 121 – 25 - ІПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

На сторінці відображається таблиця зі списком всіх товарів у магазині, із можливістю фільтрації та пошуку за різними критеріями, такими як категорія, матеріал або ціна. Також передбачена функціональність для редагування або додавання нового товару (рис. 3.8).

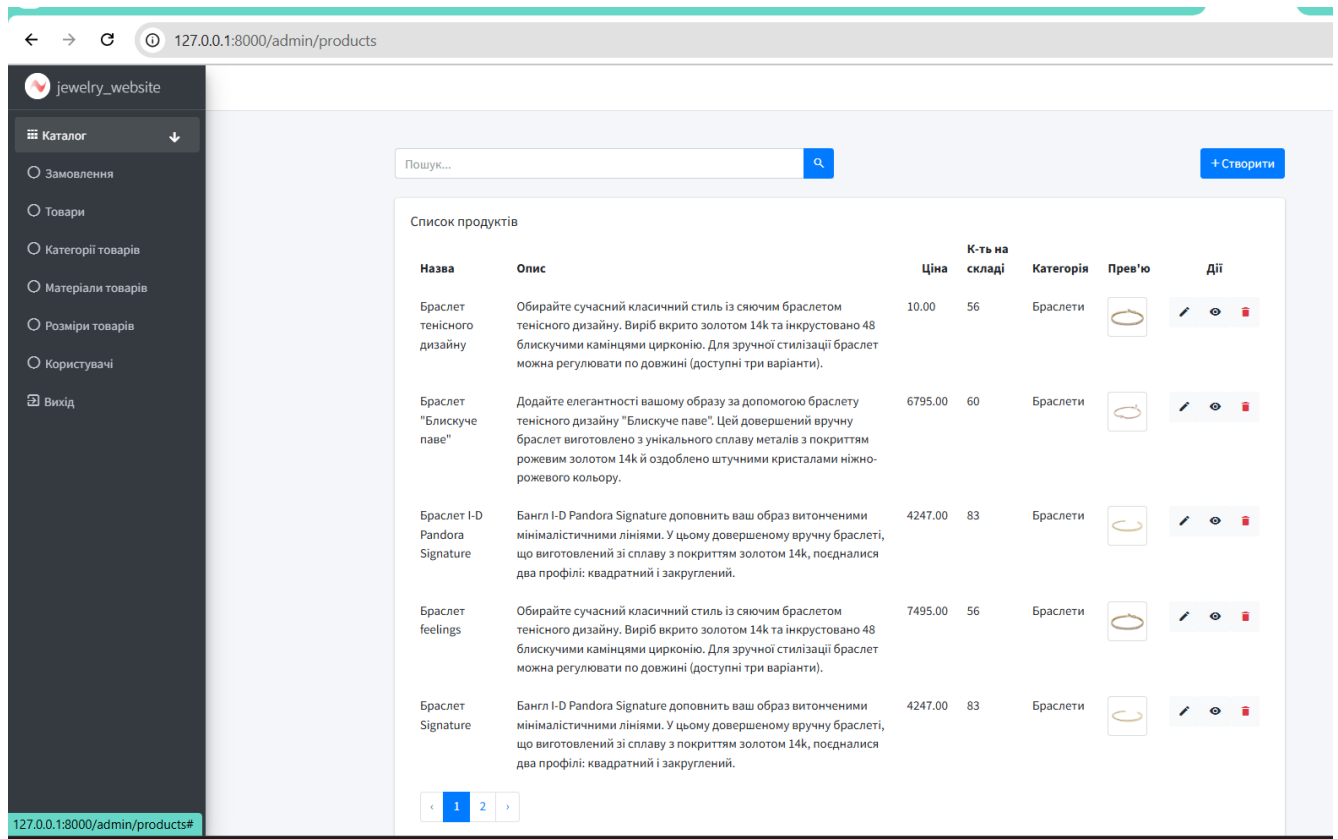


Рис. 3.8 - Сторінка управління товарами для адміна та модератора

Інтерфейс цієї сторінки розроблений таким чином, щоб адміністратори могли швидко знаходити потрібну інформацію, редагувати її та підтримувати актуальність даних у системі.

Таким чином, структура інтерфейсу забезпечує простоту і зручність у використанні на всіх етапах взаємодії з користувачем, від авторизації до підтвердження замовлення та оплати.

3.3. Тестування роботи програмного комплексу

Тестування програмного комплексу інтернет-магазину ювелірних прикрас є невід’ємним етапом життєвого циклу розробки ПЗ, що дозволяє переконатись у коректності реалізації функціоналу, стабільності роботи системи, а також зручності використання. Основну увагу було приділено ручному тестуванню інтерфейсу [20],

тестуванню функціоналу кошика, оформлення замовлення, перевірці прав доступу відповідно до ролей користувачів, а також роботі платіжного модуля LiqPay (табл. 3.1).

У ході тестування було перевірено наступні сценарії:

Таблиця 3.1

Результати тестування основних сценаріїв

	1	2	3	4
Назва сценарію	Реєстрація та логін користувача	Додавання товару до кошика	Оформлення замовлення	Оплата замовлення через LiqPay
Передумови	Відсутній акаунт	Авторизований користувач	Заповнений кошик	Замовлення створене
Кроки	Перейти на сторінку /register, заповнити форму, підтвердити e-mail	Перейти на сторінку товару, натиснути кнопку «Додати до кошика»	Перейти на /checkout, ввести адресу, натиснути «Оплатити»	На сторінці LiqPay ввести дані картки, підтвердити платіж
Очікуваний результат	Користувача зареєстровано та перенаправлено в профіль	Товар додано до кошика	Перенаправлення на сторінку LiqPay	Повідомлення «Дякуємо за покупку»
Фактичний результат	Успішно	Успішно	Успішно	Успішно
Коментар	-	-	-	Сторінка підтвердження з'явилась

Під час тестування використовувались інструменти DevTools [24] браузера для відстеження помилок у консолі, інспекції мережевих запитів, а також перевірки адаптивності інтерфейсу. Було виявлено кілька незначних проблем, таких як відсутність валідації в полях форм, які були оперативно виправлені. Це є важливою частиною процесу тестування, що дозволяє забезпечити правильність роботи інтерфейсу на різних етапах взаємодії користувача з системою [23].

Також тестуванню піддавалися окремі компоненти системи, зокрема: система аутентифікації (реєстрація, логін), інтерфейс кошика (додавання, редагування, видалення товарів), логіка збереження замовлень, інтеграція з платіжною системою LiqPay. Тестування кожної частини системи проводилося відповідно до принципів, зазначених у Laravel Testing Docs [12], що дозволяє оптимізувати процес тестування та забезпечити високу якість коду. Важливу роль у тестуванні відіграє перевірка кожного сценарію на предмет правильного виконання бізнес-логіки і взаємодії між компонентами.

Результати тестування підтверджують, що система коректно працює на всіх етапах взаємодії користувача, від реєстрації до оформлення замовлення. Під час перевірки функціональності LiqPay система виявила високу стабільність при оплаті через API, що також було підтверджено в OWASP Testing Guide v4 [11], де надано рекомендації щодо тестування платіжних систем для безпеки.

Важливим етапом також стало тестування MySQL Server [14], що дозволило забезпечити правильне підключення до бази даних та безперервне збереження замовлень, а також запитів до даних у реальному часі. Тестування MySQL також допомогло виявити деякі незначні оптимізаційні помилки у запитах, що були оперативно виправлені.

Тестування інтерфейсу було виконано із використанням GitHub Getting Started [15], що дозволило автоматизувати процес відстеження змін у коді, що важливо для збереження стабільності при внесенні змін до основного коду та запуску нових версій застосунку.

Завдяки використанню Laravel Blade Templating [13] для відображення клієнтської частини, було легко здійснити тестування рендерингу шаблонів, що гарантує правильне відображення даних на всіх етапах взаємодії з користувачем. Це допомогло у виправленні помилок рендерингу, зокрема при додаванні товарів до кошика.

Усі критичні функції працюють відповідно до поставлених вимог, а результати тестування підтверджують готовність системи до експлуатації в

					ІПЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

реальних умовах. Тестування проводилося згідно з рекомендаціями з офіційної документації Laravel, а також загальними принципами тестування веб-додатків [10].

Висновки до третього розділу

У третьому розділі було здійснено розгортання та налаштування всіх компонентів розробленої системи інтернет-магазину ювелірних прикрас. Зокрема, налаштовано серверну частину на базі Laravel, використано Blade для відображення клієнтської частини, а також налаштовано базу даних MySQL для збереження всіх даних про продукти, замовлення та користувачів. Надано детальну інструкцію з інсталяції необхідних залежностей, конфігурації середовища та запуску системи.

Було розглянуто структуру інтерфейсу веб-застосунку, де для кожної з ролей (адміністратор, модератор, користувач) визначено доступний функціонал [21]. Окремо представлені інтерфейси для реєстрації та авторизації користувачів [17], перегляду товарів, додавання товарів до кошика [16], оформлення замовлення та здійснення оплати через LiqPay. Зазначено важливість адаптивності інтерфейсу, що гарантує коректне відображення сторінок як на великих екранах, так і на мобільних пристроях, що забезпечує зручність у взаємодії з системою.

Тестування системи включало перевірку основних функцій, таких як реєстрація та авторизація користувача, додавання товарів до кошика, оформлення замовлення, а також інтеграцію з платіжною системою LiqPay. Було перевірено коректність роботи REST-запитів, генерацію сповіщень та обробку платіжних даних. Усі виявлені проблеми були оперативно усунуті, що підтверджує надійність та стабільність розробленої системи [22] .

В результаті виконаних робіт система продемонструвала свою готовність до використання в реальному середовищі, забезпечуючи ефективне керування процесами покупки, оплати і обробки замовлень. Тестування підтвердило, що всі функції працюють відповідно до вимог, що дозволяє рекомендувати систему до впровадження в реальних умовах.

					ІПЗ.КР.Б – 121 – 25 - ІЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було спроектовано та реалізовано веб-застосунок для інтернет-магазину ювелірних прикрас, що дозволяє ефективно керувати каталогом товарів, здійснювати покупки, оформляти замовлення та здійснювати оплату через платіжні системи. Система реалізована на базі фреймворку Laravel, з використанням Blade для відображення клієнтської частини.

Основну увагу було приділено розробці функціональних можливостей для кожної з ролей користувачів: адміністратора, модератора та кінцевого користувача. Було спроектовано зручний інтерфейс для перегляду товарів, додавання їх до кошика, оформлення замовлень та здійснення оплати. У результаті розробки була створена база даних, що забезпечує ефективне зберігання інформації про продукти, замовлення та користувачів.

У ході роботи було проаналізовано та визначено основні вимоги до системи, зокрема до безпеки, продуктивності та зручності для користувача. Окремо було реалізовано систему аутентифікації користувачів, що дозволяє користувачам реєструватися та входити в систему за допомогою підтверджених облікових записів.

Тестування системи показало, що усі основні функціональні можливості працюють стабільно та без помилок. Виявлені незначні недоліки були оперативно виправлені, що підтверджує готовність системи до використання в реальних умовах.

Таким чином, розроблений інтернет-магазин є функціональним, безпечним та зручним для користувачів. Система відповідає вимогам до ефективності, продуктивності та безпеки, що дозволяє рекомендувати її для впровадження в реальному бізнес-середовищі.

					ІПЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Laravel Documentation [Електронний ресурс] – Режим доступу:
<https://laravel.com/docs>
2. PHP Documentation [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.php.net/docs.php>
3. MariaDB Documentation [Електронний ресурс] – Режим доступу:
<https://mariadb.org/documentation/>
4. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
5. Freeman, E., & Robson, E. (2004). Head First Design Patterns. O'Reilly Media.
6. LiqPay API Documentation [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.liqpay.ua/documentation/en>
7. Composer Documentation [Електронний ресурс] – Режим доступу:
<https://getcomposer.org/doc/>
8. Git Documentation [Електронний ресурс] – Режим доступу до ресурсу:
<https://git-scm.com/doc>
9. Spatie Laravel Permission [Електронний ресурс] – Режим доступу до ресурсу:
<https://spatie.be/docs/laravel-permission>
10. Software Testing Fundamentals [Електронний ресурс] – Режим доступу до ресурсу:
<https://softwaretestingfundamentals.com/>
11. OWASP Testing Guide v4 [Електронний ресурс] – Режим доступу до ресурсу:
<https://owasp.org/www-project-web-security-testing-guide/>
12. Laravel Testing Docs [Електронний ресурс] – Режим доступу до ресурсу:
<https://laravel.com/docs/testing>
13. Laravel Blade Templating [Електронний ресурс] – Режим доступу до ресурсу:
<https://laravel.com/docs/blade>
14. MySQL Server Installation Guide [Електронний ресурс] – Режим доступу до ресурсу:
<https://dev.mysql.com/doc/refman/8.0/en/installing.html>

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

15. GitHub Getting Started [Електронний ресурс] – Режим доступу до ресурсу:
<https://docs.github.com/en/get-started/quickstart>
16. How to Implement Shopping Cart in Laravel – Режим доступу: <https://laravel-news.com/creating-shopping-cart-laravel>
17. Laravel Authentication Documentation – Режим доступу:
<https://laravel.com/docs/8.x/authentication>
18. Access Control in Laravel [Електронний ресурс] – Режим доступу до ресурсу:
<https://laravel.com/docs/10.x/authorization>
19. Non-Functional Requirements in Software Engineering [Електронний ресурс] –
Режим доступу до ресурсу: <https://www.sans.edu/cybersecurity/blog/non-functional-requirements>
20. Manual Testing for Web Applications] – Режим доступу:
<https://www.softwaretestinghelp.com/manual-testing/>
21. Responsive Web Design Basics – Режим доступу: <https://web.dev/responsive-web-design-basics/>
22. REST API Best Practices [Електронний ресурс] – Режим доступу до ресурсу:
<https://restfulapi.net/>
23. Web Performance Optimization – Режим доступу:
<https://www.keycdn.com/blog/web-performance-testing>
24. DevTools for Testing Web Applications – Режим доступу:
<https://developer.chrome.com/docs/devtools/>
25. Role-Based Access Control (RBAC) in Web Applications – Режим доступу:
<https://www.digitalocean.com/community/tutorials>
26. DBMS [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.techopedia.com/definition/23983/database-management-system-dbms>
27. Docker Documentation [Електронний ресурс] – Режим доступу до ресурсу:
<https://docs.docker.com/>

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

28.PostgreSQL Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/docs/>

29.MongoDB [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mongodb.com/>

30.CRUD Operations in Laravel - A Comprehensive Guide [Електронний ресурс] – Режим доступу: <https://www.tutsmake.com/crud-operations-in-laravel/>

					ІПЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг контролеру OrderController:

```

<?php

namespace App\Http\Controllers\Front;

use App\Http\Controllers\Controller;
use App\Models\Cart;
use App\Models\Order;
use App\Services\LiqPayService;
use Illuminate\Contracts\View\Factory;
use Illuminate\Foundation\Application;
use Illuminate\Support\Facades\Auth;
use Illuminate\Http\Request;
use App\Http\Requests\OrderStoreRequest;
use Illuminate\Http\RedirectResponse;

class OrderController extends Controller
{
    public function create(Request $request)
    {
        $cart = Cart::where('user_id', auth()->id()->first();

        if (!$cart || $cart->products->isEmpty()) {
            return redirect()->route('carts.index')->with('error', 'Ваш кошик порожній!');
        }

        return view('front.pages.checkout', compact('cart'));
    }

    public function checkout(Request $request)
    {
        $cart = Cart::where('user_id', Auth::id()->first();

        if (!$cart || $cart->products->isEmpty()) {
            return redirect()->route('carts.index')->with('error', 'Ваш кошик порожній.');

```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return $product->price * $product->pivot->quantity;
    }
    ),
    'shipping_address' => $request->shipping_address,
    'shipping_city' => $request->shipping_city,
    'shipping_postal_code' => $request->shipping_postal_code,
    'shipping_country' => $request->shipping_country,
    'phone' => $request->phone,
    'notes' => $request->notes,
    'placed_at' => now(),
]);

$cart = Cart::with('products')->where('user_id', auth()->id())->first();

foreach ($cart->products as $product) {
    $order->products()->attach($product->id, [
        'quantity' => $product->pivot->quantity,
        'price' => $product->price,
    ]);
}

return redirect()->route('checkout.payment', $order)->with('success', 'Замовлення створено!');
}

public function liqpayCallback(Request $request, LiqPayService $liqPayService)
{
    $data = $request->input('data');
    $response = $liqPayService->parseCallback($data);

    if (isset($response['order_id']) && $response['status'] === 'success') {
        $order = Order::find($response['order_id']);
        if ($order) {
            $order->payment_status = 'paid';
            $order->status = 'paid';
            $order->save();

            $cart = Cart::where('user_id', $order->user_id)->first();
            if ($cart) {
                $cart->products()->detach();
            }
        }
    }

    return response('OK', 200);
}

public function payment(Order $order, LiqPayService $liqPayService)
{
    $form = $liqPayService->getForm([
        'action' => 'pay',
        'amount' => $order->total_price,
        'currency' => 'UAH',
        'description' => 'Замовлення №' . $order->id,
        'order_id' => $order->id,
        'version' => 3,
        'language' => 'uk',
        'result_url' => route('checkout.success', $order),
    ]);
}

```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		63

```

        'server_url' => route('checkout.callback', $order),
    ]);

    return view('front.pages.payment', compact('form', 'order'));
}

public function paymentSuccess(Order $order):
Factory|Application\Illuminate\Contracts\View\View|Illuminate\Contracts\Foundation\Application
{
    return view('front.pages.payment-success', compact('order'));
}
}

```

Лістинг контролеру ProductController:

```

<?php

namespace App\Http\Controllers;

use App\Models\Product;
use Illuminate\View\View;
use Illuminate\Http\Request;
use App\Models\ProductCategory;
use Illuminate\Http\RedirectResponse;
use Illuminate\Support\Facades\Storage;
use App\Http\Requests\ProductStoreRequest;
use App\Http\Requests\ProductUpdateRequest;

class ProductController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index(Request $request): View
    {
        $this->authorize('viewAny', Product::class);

        $search = $request->get('search', '');

        $products = Product::search($search)
            ->latest()
            ->paginate(5)
            ->withQueryString();

        return view('app.products.index', compact('products', 'search'));
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create(Request $request): View
    {
        $this->authorize('create', Product::class);

        $productCategories = ProductCategory::pluck('name', 'id');
    }
}

```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    return view('app.products.create', compact('productCategories'));
}

/**
 * Store a newly created resource in storage.
 */
public function store(ProductStoreRequest $request): RedirectResponse
{
    $this->authorize('create', Product::class);

    $validated = $request->validated();

    $product = Product::create($validated);

    if ($request->hasFile('preview')) {
        $product->preview = $request->file('preview')->store('public/products/' . $product->id . '/preview');
    }

    if ($request->hasFile('gallery')) {
        $gallery = [];
        foreach ($request->file('gallery') as $file) {
            $gallery[] = $file->store('public/products/' . $product->id . '/gallery');
        }
        $product->gallery = json_encode($gallery);
    }

    $product->save();

    return redirect()
        ->route('products.edit', $product)
        ->withSuccess(__('crud.common.created'));
}

/**
 * Display the specified resource.
 */
public function show(Request $request, Product $product): View
{
    $this->authorize('view', $product);

    return view('app.products.show', compact('product'));
}

/**
 * Show the form for editing the specified resource.
 */
public function edit(Request $request, Product $product): View
{
    $this->authorize('update', $product);

    $productCategories = ProductCategory::pluck('name', 'id');

    return view(
        'app.products.edit',

```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        compact('product', 'productCategories')
    );
}

/**
 * Update the specified resource in storage.
 */
public function update(ProductUpdateRequest $request, Product $product): RedirectResponse
{
    $this->authorize('update', $product);

    $validated = $request->validated();

    if ($request->hasFile('preview')) {
        if ($product->preview) {
            Storage::delete('public/products/' . $product->id . '/preview/' . $product->preview);
        }

        $validated['preview'] = $request->file('preview')->store('public/products/' . $product->id . '/preview');
    }

    if ($request->hasFile('gallery')) {
        if ($product->gallery) {
            foreach (json_decode($product->gallery) as $file) {
                Storage::delete('public/products/' . $product->id . '/gallery/' . $file);
            }
        }

        $gallery = [];
        foreach ($request->file('gallery') as $file) {
            $gallery[] = $file->store('public/products/' . $product->id . '/gallery');
        }
        $validated['gallery'] = json_encode($gallery);
    }

    $product->update($validated);

    return redirect()
        ->route('products.edit', $product)
        ->withSuccess(__('crud.common.saved'));
}

/**
 * Remove the specified resource from storage.
 */
public function destroy(
    Request $request,
    Product $product
): RedirectResponse
{
    $this->authorize('delete', $product);

    if ($product->preview) {
        Storage::delete($product->preview);
    }
}

```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if ($product->gallery) {
            Storage::delete($product->gallery);
        }

        $product->delete();

        return redirect()
            ->route('products.index')
            ->withSuccess(__('crud.common.removed'));
    }

    public function deleteGalleryImage(Request $request, Product $product)
    {
        $image = $request->input('image');
        $gallery = json_decode($product->gallery, true);

        if (($key = array_search($image, $gallery)) !== false) {
            Storage::delete('public/products/' . $product->id . '/gallery/' . $image);
            unset($gallery[$key]);
            $product->gallery = json_encode(array_values($gallery));
            $product->save();
        }

        return response()->json(['success' => true]);
    }
}

```

Лістинг контролеру UserController:

```

<?php

namespace App\Http\Controllers;

use App\Models\User;
use Illuminate\View\View;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Http\RedirectResponse;
use App\Http\Requests\UserStoreRequest;
use App\Http\Requests\UserUpdateRequest;

class UserController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index(Request $request): View
    {
        $this->authorize('view-any', User::class);

        $search = $request->get('search', '');

        $users = User::search($search)
            ->latest()

```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

->paginate(5)
->withQueryString();

return view('app.users.index', compact('users', 'search'));
}

/**
 * Show the form for creating a new resource.
 */
public function create(Request $request): View
{
    $this->authorize('create', User::class);

    return view('app.users.create');
}

/**
 * Store a newly created resource in storage.
 */
public function store(UserStoreRequest $request): RedirectResponse
{
    $this->authorize('create', User::class);

    $validated = $request->validated();

    $validated['password'] = Hash::make($validated['password']);

    $user = User::create($validated);

    return redirect()
        ->route('users.edit', $user)
        ->withSuccess(__('crud.common.created'));
}

/**
 * Display the specified resource.
 */
public function show(Request $request, User $user): View
{
    $this->authorize('view', $user);

    return view('app.users.show', compact('user'));
}

/**
 * Show the form for editing the specified resource.
 */
public function edit(Request $request, User $user): View
{
    $this->authorize('update', $user);

    return view('app.users.edit', compact('user'));
}

/**
 * Update the specified resource in storage.

```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    */
    public function update(
        UserUpdateRequest $request,
        User $user
    ): RedirectResponse {
        $this->authorize('update', $user);

        $validated = $request->validated();

        if (empty($validated['password'])) {
            unset($validated['password']);
        } else {
            $validated['password'] = Hash::make($validated['password']);
        }

        $user->update($validated);

        return redirect()
            ->route('users.edit', $user)
            ->withSuccess(__('crud.common.saved'));
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy(Request $request, User $user): RedirectResponse
    {
        $this->authorize('delete', $user);

        $user->delete();

        return redirect()
            ->route('users.index')
            ->withSuccess(__('crud.common.removed'));
    }
}

```

Лістинг контролеру ProductMaterialController:

```

<?php

namespace App\Http\Controllers;

use Illuminate\View\View;
use Illuminate\Http\Request;
use App\Models\ProductMaterial;
use Illuminate\Http\RedirectResponse;
use App\Http\Requests\ProductMaterialStoreRequest;
use App\Http\Requests\ProductMaterialUpdateRequest;

class ProductMaterialController extends Controller
{
    /**
     * Display a listing of the resource.
     */
}

```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

public function index(Request $request): View
{
    $this->authorize('view-any', ProductMaterial::class);

    $search = $request->get('search', '');

    $productMaterials = ProductMaterial::search($search)
        ->latest()
        ->paginate(5)
        ->withQueryString();

    return view(
        'app.product_materials.index',
        compact('productMaterials', 'search')
    );
}

/**
 * Show the form for creating a new resource.
 */
public function create(Request $request): View
{
    $this->authorize('create', ProductMaterial::class);

    return view('app.product_materials.create');
}

/**
 * Store a newly created resource in storage.
 */
public function store(
    ProductMaterialStoreRequest $request
): RedirectResponse {
    $this->authorize('create', ProductMaterial::class);

    $validated = $request->validated();

    $productMaterial = ProductMaterial::create($validated);

    return redirect()
        ->route('product-materials.edit', $productMaterial)
        ->withSuccess(__('crud.common.created'));
}

/**
 * Display the specified resource.
 */
public function show(
    Request $request,
    ProductMaterial $productMaterial
): View {
    $this->authorize('view', $productMaterial);

    return view('app.product_materials.show', compact('productMaterial'));
}

```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

/**
 * Show the form for editing the specified resource.
 */
public function edit(
    Request $request,
    ProductMaterial $productMaterial
): View {
    $this->authorize('update', $productMaterial);

    return view('app.product_materials.edit', compact('productMaterial'));
}

/**
 * Update the specified resource in storage.
 */
public function update(
    ProductMaterialUpdateRequest $request,
    ProductMaterial $productMaterial
): RedirectResponse {
    $this->authorize('update', $productMaterial);

    $validated = $request->validated();

    $productMaterial->update($validated);

    return redirect()
        ->route('product-materials.edit', $productMaterial)
        ->withSuccess(__('crud.common.saved'));
}

/**
 * Remove the specified resource from storage.
 */
public function destroy(
    Request $request,
    ProductMaterial $productMaterial
): RedirectResponse {
    $this->authorize('delete', $productMaterial);

    $productMaterial->delete();

    return redirect()
        ->route('product-materials.index')
        ->withSuccess(__('crud.common.removed'));
}
}

```

Лістинг моделі Product:

```

<?php

namespace App\Models;

use App\Models\Scopes\Searchable;
use Illuminate\Database\Eloquent\Model;

```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

use Illuminate\Database\Eloquent\Factories\HasFactory;

class Product extends Model
{
    use HasFactory;
    use Searchable;

    protected $fillable = [
        'name',
        'description',
        'price',
        'preview',
        'gallery',
        'stock_quantity',
        'product_category_id',
    ];

    protected $searchableFields = ['*'];

    protected $casts = [
        'gallery' => 'array',
    ];

    public function productCategory(): \Illuminate\Database\Eloquent\Relations\BelongsTo
    {
        return $this->belongsTo(ProductCategory::class);
    }

    public function orders(): \Illuminate\Database\Eloquent\Relations\BelongsToMany
    {
        return $this->belongsToMany(Order::class, 'order_has_products');
    }

    public function productMaterials(): \Illuminate\Database\Eloquent\Relations\BelongsToMany
    {
        return $this->belongsToMany(
            ProductMaterial::class,
            'product_has_materials'
        );
    }

    /**
     * @param $value
     * @return array
     */
    public function getGalleryAttribute($value): array
    {
        $decodedValue = json_decode($value, true);

        if (json_last_error() !== JSON_ERROR_NONE) {
            return [];
        }

        if (is_string($decodedValue)) {
            $decodedValue = json_decode($decodedValue, true);
        }
    }
}

```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг заповнювача PermissionsSeeder:

					ІПЗ.КР.Б – 121 – 25 - ІПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

['name' => 'create products'],
['name' => 'update products'],
['name' => 'delete products'],

// Product Categories
['name' => 'list product categories'],
['name' => 'view product categories'],
['name' => 'create product categories'],
['name' => 'update product categories'],
['name' => 'delete product categories'],

// Product Sizes
['name' => 'list product sizes'],
['name' => 'view product sizes'],
['name' => 'create product sizes'],
['name' => 'update product sizes'],
['name' => 'delete product sizes'],

// Product Materials
['name' => 'list product materials'],
['name' => 'view product materials'],
['name' => 'create product materials'],
['name' => 'update product materials'],
['name' => 'delete product materials'],

// Cart
['name' => 'list carts'],
['name' => 'view carts'],
['name' => 'create carts'],
['name' => 'update carts'],
['name' => 'delete carts'],

];

// Create all permissions
foreach ($allPermissionsNames as $permission) {
    Permission::create($permission);
}

// Give all permissions to admin
$allPermissions = Permission::all();
$role1->givePermissionTo($allPermissions);

// Define moderator permissions
$moderatorPermissionsNames = [
    'list users', 'view users',
    'list orders', 'view orders',
    'list products', 'view products', 'create products', 'update products', 'delete products',
    'list product categories', 'view product categories', 'create product categories', 'update product categories',
    'delete product categories',
    'list product sizes', 'view product sizes', 'create product sizes', 'update product sizes', 'delete product sizes',
    'list product materials', 'view product materials', 'create product materials', 'update product materials',
    'delete product materials',
    'list carts', 'view carts',
];

```

```

// Assign moderator permissions
$moderatorPermissions = Permission::whereIn('name', $moderatorPermissionsNames)->get();
$role2->givePermissionTo($moderatorPermissions);

// Define user permissions
$userPermissionsNames = [
    'create orders',
    'list products', 'view products',
    'view product categories', 'list product categories',
    'view product sizes', 'list product sizes',
    'view product materials', 'list product materials',
];

// Assign user permissions
$userPermissions = Permission::whereIn('name', $userPermissionsNames)->get();
$role3->givePermissionTo($userPermissions);

// Assign roles to users
$admin = User::whereEmail('admin@admin.com')->first();
$moderator = User::whereEmail('moderator@moderator.com')->first();

if ($admin) {
    $admin->assignRole($role1);
}

if ($moderator) {
    $moderator->assignRole($role2);
}
}
}

```

					ІІЗ.КР.Б – 121 – 25 - ІІЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		