

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:

**«Вебплатформа для створення косметичних
колекцій та ведення блогу»**

Виконала студентка групи ІІЗ-22-1
ТРИБЮК Варвара Олександрівна

Керівник роботи:
ФАНТ Микола Олександрович

Рецензент:
ТОЛСТОЙ Ігор Анатолійович

Житомир – 2026

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна Вакалюк

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **Трибюк Варвара Олександрівна**

Керівник роботи: **ФАНТ Микола Олександрович**

Тема роботи: **«Вебплатформа для створення косметичних колекцій та ведення блогу»**,

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92**

Вихідні дані для роботи: **об'єктом дослідження є процес проєктування та розробки вебплатформи для візуальної організації косметичних засобів та публікації користувацького контенту, предметом дослідження є методи, архітектурні рішення та програмні засоби реалізації інтерактивної вебплатформи для створення косметичних колекцій та ведення блогу.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Фант М.О.	02.03.2026	02.03.2026
2	Фант М.О.	02.03.2026	02.03.2026
3	Фант М.О.	02.03.2026	02.03.2026

РЕФЕРАТ

Пояснювальна записка до випускної кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, джерел інформації та додатків.

Текстова частина викладена на 82 сторінках друкованого тексту.

Джерела інформації містять 31 найменування і займають 3 сторінки. Пояснювальна записка має 10 сторінок додатків. У роботі наведено 65 рисунків. Загальний обсяг роботи – 92 сторінка.

Метою випускної кваліфікаційної роботи бакалавра є проєктування та розробка вебплатформи для створення та управління косметичними колекціями, ведення блогу користувачів, обміну досвідом щодо використання косметичних засобів, а також інтеграції інструментів штучного інтелекту для аналізу складу косметичної продукції.

У роботі було проаналізовано аналоги, та визначено їх основні недоліки, з якими можуть зіштовхуватися користувачі під час використання. На основі цього аналізу було визначено основні функціональні та нефункціональні вимоги. Обрано засоби реалізації та архітектурні підходи для вирішення поставленої задачі. Та описано роботу веб платформи.

КЛЮЧОВІ СЛОВА: КОСМЕТИЧНІ КОЛЕКЦІЇ, ДОПИСИ, ASP.NET, REACT, AZURE, AZURE BLOB STORAGE.

					ІПЗ.КР.Б – 121 – 26 – ПЗ		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Трибюк В.О.			Вебплатформа для створення косметичних колекцій та ведення блогу Пояснювальна записка	Літ.	Арк.
Керівник.		Фант М.О.					2
						Аркуші	92
Рецензент		Толстой І.А.				Житомирська політехніка, група ІПЗ-22-1	
Зав. каф.		Вакалюк Т.А.					

ABSTRACT

The bachelor's qualification project consists of an introduction, three chapters, conclusions, references, and appendices.

The main text comprises 81 pages. The list of references contains 31 sources and occupies 3 pages. The explanatory note includes 10 pages of appendices. The thesis contains 65 figures. The total volume of the thesis is 91 pages.

The purpose of the bachelor's qualification thesis is to design and develop a web platform for creating and managing cosmetic collections, maintaining user blogs, sharing experience regarding cosmetic products, and integrating artificial intelligence tools for cosmetic ingredient analysis.

The thesis analyzes existing solutions and identifies their main shortcomings that users may encounter during use. Based on this analysis, the functional and non-functional requirements of the system were defined. Appropriate implementation technologies and architectural approaches were selected to solve the stated problem. The operation and functionality of the web platform are also described.

KEYWORDS: COSMETIC COLLECTIONS, POSTS, ASP.NET, REACT, AZURE, AZURE BLOB STORAGE.

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

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ЗМІСТ	5
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	7
ВСТУП	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ВЕБПЛАТФОРМИ ДЛЯ СТВОРЕННЯ КОСМЕТИЧНИХ КОЛЕКЦІЙ ТА ВЕДЕННЯ БЛОГУ	10
1.1 Постановка задачі	10
1.2 Аналіз аналогів програмного продукту	13
1.3 Вибір архітектури для вебплатформи створення косметичних колекцій та ведення блогу	20
1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	23
Висновки до першого розділу.....	26
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБПЛАТФОРМИ ДЛЯ СТВОРЕННЯ КОСМЕТИЧНИХ КОЛЕКЦІЙ ТА ВЕДЕННЯ БЛОГУ	28
2.1 Визначення варіантів використання та об’єктно-орієнтованої структури системи до вебплатформи	28
2.2 Розробка бази даних системи.....	39
2.3 Проєктування та реалізація алгоритмів роботи системи	43
2.4 Реалізація вебплатформи для створення косметичних колекцій та ведення блогу	46
Висновки до другого розділу.....	51
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ДЛЯ СТВОРЕННЯ КОСМЕТИЧНИХ КОЛЕКЦІЙ ТА ВЕДЕННЯ БЛОГУ	53
3.1 Порядок встановлення та налаштування параметрів	53

3.2 Структура інтерфейсу вебплатформи для створення косметичної колекції та ведення блогу	56
3.3 Тестування роботи програмного комплексу	74
Висновки до третього розділу	77
ВИСНОВКИ.....	78
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	80
ДОДАТКИ.....	83

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – база даних

JSON – JavaScript Object Notation.

REST API – Representational State Transfer Application Programming Interface.

HTTP – HyperText Transfer Protocol.

ORM – Object-Relational Mapping.

SQL – Structured Query Language.

LINQ – Language Integrated Query.

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

ВСТУП

Актуальність теми полягає в тому, що косметичні продукти стали невід’ємною частиною повсякденного життя сучасної людини. Активний розвиток соціальних мереж суттєво впливає на формування споживчих уподобань у beauty-сфері.

Щодня користувачі натрапляють на велику кількість інформації у вигляді відеооглядів, рекомендацій щодо догляду за шкірою обличчя, волоссям, тілом тощо. Це формує середовище постійного пошуку нових і «кращих» засобів, що часто призводить до хаотичності у власних бажаннях та надлишку косметичних продуктів.

За аналітичними прогнозами, у 2026 році світовий ринок косметики та засобів особистої гігієни генеруватиме дохід у розмірі 698,38 млрд доларів США [1]. Через це на ринку супроводжується постійна поява нових косметичних брендів, які за допомогою різних засобів прагнуть захопити увагу людей.

Саме ці умови створюють підґрунтя для розробки спеціалізованої платформи, де користувачі можуть не лише споживати контент, а й формувати власні косметичні колекції. Такий підхід дозволяє структурувати отриману інформацію у вигляді колекцій продуктів, орієнтованих на різні потреби. Окрім функції органайзера, важливим є соціальний аспект – можливість ведення блогу та обговорення тем, пов’язаних із beauty-індустрією, що сприяє обміну реальним досвідом і формуванню більш усвідомленого вибору косметичних засобів.

Метою випускної кваліфікаційної роботи бакалавра є проєктування та розробка вебплатформи яка призначена для зручної візуальної організації косметичних засобів у колекції з вбудованими можливостями для ведення блогу.

Основними завданнями є:

1. Пошук та аналіз існуючих аналогів програмного продукту.
2. Формування вимог до вебплатформи на основі проведеного аналізу.
3. Аналіз та вибір технологій та засобів розробки.
4. Розробка архітектури програмного забезпечення.

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

5. Програмна реалізація вебплатформи, використовуючи вибраний стек технологій.

6. Проведення тестування вебплатформи, на відповідність до поставлених вимог.

Об’єктом дослідження є процес проєктування та розробки вебплатформи для візуальної організації косметичних засобів та публікації користувацького контенту.

Предметом дослідження є методи, архітектурні рішення та програмні засоби реалізації інтерактивної вебплатформи для створення косметичних колекцій та ведення блогу.

Практичне значення отриманих результатів полягає у створенні вебплатформи, яка надає користувачам можливість організовувати власні косметичні колекції, вести блог, взаємодіяти з іншими учасниками спільноти та отримувати інформацію про косметичні засоби. Розроблений програмний продукт може бути використаний як основа для подальшого розвитку спеціалізованих соціальних платформ у індустрії краси.

Апробація результатів роботи. За темою випускної кваліфікаційної роботи бакалавра було опубліковано тези:

Трибюк В.О., Фант М.О. Стек технологій для реалізація платформи для косметичних колекцій та блогу. Тези доповідей VIII Всеукраїнської науково-технічної конференції «Комп’ютерні технології: інновації, проблеми, рішення», 02-03 грудня 2025 р. Житомир: Державний університет «Житомирська політехніка», 2025. С. 33–34.

Трибюк В.О., Фант М.О. Реалізація поведінкового шаблону Стратегія для обробки взаємодій користувачів з об’єктами системи. Тези доповідей XI Міжнародної науково-технічної конференції «Інформаційно-комп’ютерні технології», 02-03 квітня 2026 р. Житомир: Державний університет «Житомирська політехніка», 2026. С. 35-36.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ВЕБПЛАТФОРМИ ДЛЯ СТВОРЕННЯ КОСМЕТИЧНИХ КОЛЕКЦІЙ ТА ВЕДЕННЯ БЛОГУ

1.1 Постановка задачі

Сучасний цифровий простір генерує постійний потік інформації, що нерідко призводить до перенасичення. Користувачі намагаються запам'ятати цікаві продукти або організувати їх, використовуючи підручні засоби, наприклад, звичайні нотатки в телефоні чи електронні таблиці.

Однак такі підходи є малоефективними та незручними, оскільки їх налаштування потребує значних витрат часу та енергії, тоді як більшість користувачів очікує швидкої та легкої взаємодії без додаткового навантаження.

Окрім функціональності, важливу роль відіграє емоційна складова у взаємодії з застосунками. Людське прагнення до естетики та впорядкованості є природнім бажанням. Користувачам важливо не просто зберігати інформацію, а отримувати візуальне задоволення від взаємодії з власною колекцією, що дозволяє відчувати контроль та гармонію, коли все необхідне зібрано в єдиному, зручному просторі.

Для визначення основних функціональних можливостей системи необхідно проаналізувати проблеми, з якими стикаються користувачі під час взаємодії з косметичними продуктами та інформацією про них.

По-перше, як було зазначено вище, значною проблемою є нестача зручних платформ для створення та ведення персональних косметичних колекцій. По-друге, важливу роль відіграє естетичний аспект, а саме платформа має бути інтуїтивно зрозумілою, візуально привабливою та зручною у використанні. По-третє, актуальним є соціальний аспект, що передбачає можливість обміну досвідом, порадами та рекомендаціями між користувачами.

Сукупність визначених проблем стала підґрунтям для розробки програмного продукту, який спрямований на усунення виявлених недоліків існуючих рішень та задоволення потреб користувачів у сфері організації косметичних засобів і взаємодії в межах спільноти.

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

Оскільки платформа має соціальну спрямованість, важливим аспектом її функціонування є створення персонального профілю користувача. Користувач потребує можливість налаштовувати інформацію про себе, додавати опис та теги, що відображають його інтереси й уподобання. Механізми керування приватністю профілю, можуть надати змогу обмежувати доступ до опублікованого контенту лише для підписників, що забезпечує можливість формування власної спільноти між користувачами та контроль доступу до особистого контенту.

Продовжуючи напрям соціальної взаємодії, платформа передбачатиме можливість перегляду контенту інших учасників та коментарі і реакції до них. Це сприятиме формуванню взаємодії між користувачами та створенню дружнього середовища в межах системи.

Основною метою платформи є впорядкування косметичних продуктів користувачів, тому створення власних колекцій та їх об'єднання у тематичні розділи дозволяє структурувати інформацію відповідно до особистих потреб і забезпечує більш зручне керування косметичними засобами. Реалізація функціональності збереження косметичних колекцій інших користувачів до розділу «Вибране» дозволить формувати персональні добірки контенту та повертатися до них у подальшому, що сприяє активнішій взаємодії з контентом інших учасників.

Оскільки основними об'єктами платформи є косметичні засоби, потрібно реалізувати можливості їхнього пошуку, фільтрації за їхніми особливостями, оцінками, категоріями чи брендами. Через те що неможливо на платформу додати всі можливі косметичні продукти, повинна бути передбачена можливість додавання їх у систему, а саме, на вимогу, тобто учасник платформи створює запит, а адміністрація сайту розглядає та обробляє цей запит.

Також доцільним є використання технологій штучного інтелекту для аналізу складу косметичних засобів. Оскільки багато користувачів не розуміють, що саме міститься у косметичних продуктах, це створює проблему необґрунтованого вибору та придбання «всліпу». Тому на основі аналізу

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

інгредієнтів та категорії продукту система може формувати узагальнену інформацію, щодо потенційних переваг, можливих ризиків. Це допоможе користувачам приймати більш обґрунтовані рішення під час вибору косметичних засобів.

Враховуючи те, що платформа передбачає можливість ведення блогу, реалізація текстового редактору, забезпечить не лише створення текстового контенту, а й дозволить додавання інших об'єктів платформи. Автори повинні мати можливість створювати, редагувати, архівувати та видаляти власні публікації. Для підвищення інформативності постів редактор повинен підтримувати додавання зображень, косметичних продуктів, нотаток і колекцій користувачів, а також позначати користувачів у пості.

Після формування вимог до системи необхідно спроектувати структуру бази даних, яка забезпечуватиме надійне зберігання інформації про користувачів, косметичних продуктів, колекцій, публікацій та інших об'єктів системи. При цьому база даних повинна бути достатньо гнучкою для подальшого розширення функціональності платформи.

Важливим завданням також є вибір архітектурного підходу, який забезпечить масштабованість, підтримуваність та розширюваність програмного продукту. Архітектура системи повинна сприяти чіткому розподілу відповідальностей між компонентами, спрощувати тестування та подальший супровід застосунку.

Окрему увагу необхідно приділити розробці зручного та естетично привабливого користувацького інтерфейсу. Оскільки у вище згаданій проблемі є візуальна організація косметичних колекцій, інтерфейс повинен бути інтуїтивно зрозумілим, привабливим та комфортним у використанні. Крім того, вебплатформа повинна коректно працювати на пристроях із різними розмірами екранів, що потребує реалізації адаптивного дизайну.

Майбутня реалізація має не лише спростити повсякденне використання платформи для звичайних користувачів, а й надати інструменти для більш професійного застосування. Це дозволить візажистам, мейкап-кріейторам та

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

іншим фахівцям ефективно формувати й організовувати колекції продуктів для відеооглядів, рекомендацій та професійної діяльності, а також ділитися експертним досвідом із широкою аудиторією. Крім того, платформа сприятиме підвищенню обізнаності користувачів щодо косметичних засобів завдяки, можливості обміну досвідом та можливості їх аналізу в одному середовищі.

1.2 Аналіз аналогів програмного продукту

Проведення аналізу існуючих аналогів є важливим етапом для майбутнього проектування програмного продукту, який дозволяє не лише оцінити наявні на ринку рішення, а й виявити їхні слабкі сторони. Це дає розуміння того, з якими потенційними проблемами може зіштовхнутися користувач, під час використання певного продукту та що можна покращити вже в існуючій системі для більш зручнішого використання.

Для визначення вимог до майбутнього програмного продукту було проведено детальний аналіз популярних платформ, які частково задовольняють потреби користувачів.

Одним із найвідоміших подібних рішень є платформа Pinterest [2]. Це соціальний вебсервіс, основна мета якого полягає у створенні тематичних дошок (рис. 1.1).

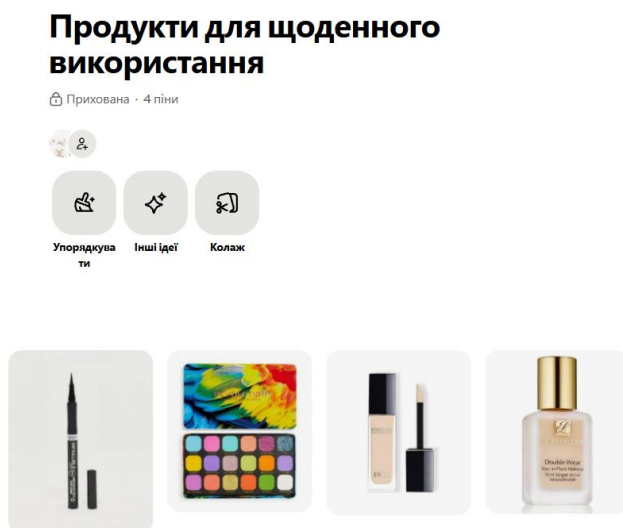


Рис. 1.1. Вигляд власної дошки

Користувачі там можуть зберігати зображення чи відео, так звані «піни», відповідно до обраної тематики, а також зберігати ці дошки у відповідних тематичних розділах (рис. 1.2).

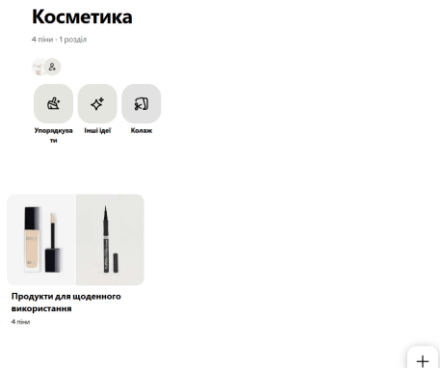


Рис. 1.2. Створена дошка у тематичному розділі

За своєю суттю Pinterest виконує роль джерела для натхнення та інструмента візуалізації ідей, однак у контексті керування косметичною колекцією має низку проблем та обмежень. По-перше, у разі бажання створити детальний опис косметичного продукту користувач змушений заповнювати всю інформацію вручну (рис. 1.3), в тому числі опис, властивості та особливості використання продукту.

Рис. 1.3. Вигляд створення власного піну

По-друге, така модель організації є недостатньо гнучкою, інформація про продукт не адаптується під особливості інших користувачів. Наприклад, у випадку, коли споживач має інший відтінок одного й того ж засобу, він не має

можливості змінити інформацію, щоб персоналізувати наявні дані відповідно до власних потреб (рис. 1.4).

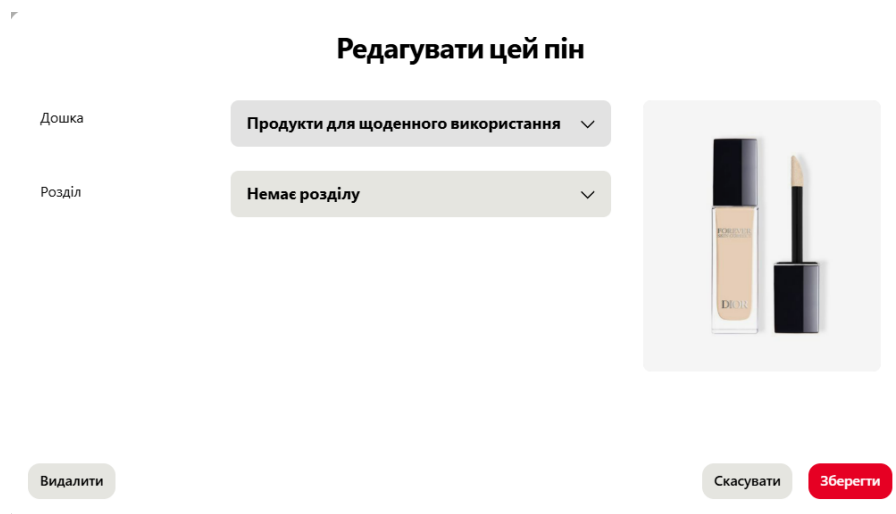


Рис. 1.4. Редагування піну

По-третє, платформа не передбачає можливості коментування дошок інших користувачів, взаємодія доступна лише тільки у випадку, якщо користувач доданий до дошки як учасник, де він може спілкуватися з іншими членами в чаті (рис. 1.5).

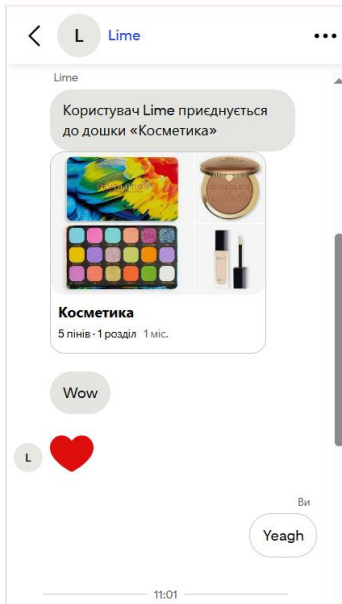


Рис. 1.5. Спільний чат з учасниками дошки

Наступним аналогом є Sephora Community [3] – онлайн-спільнота на базі великого рітейлера. Даний ресурс надає широкі можливості для соціальних взаємодій, можна залишати відгуки, ставити питання різним активним користувачам та обговорювати продукти на форумі (рис. 1.6).

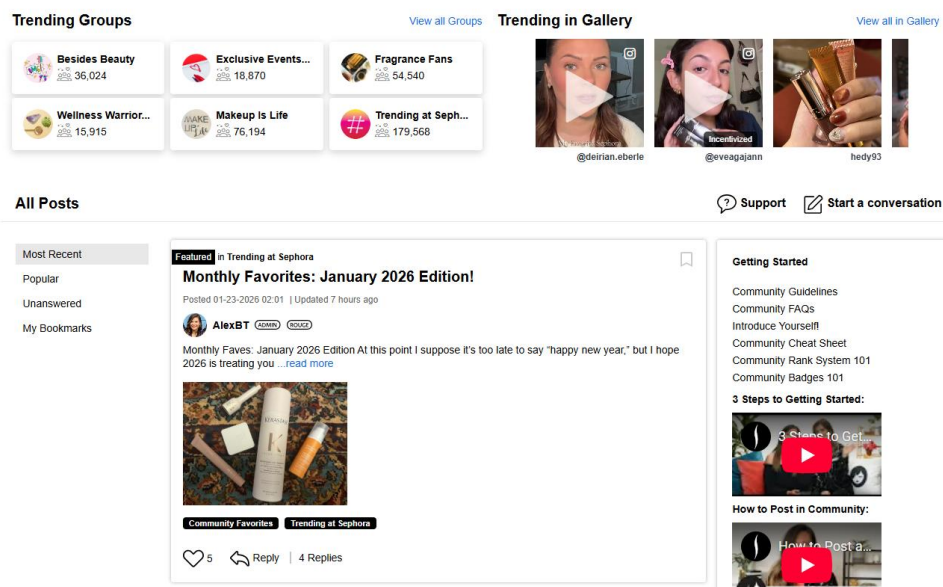


Рис. 1.6. Головна сторінка сайту

Основною особливістю створення косметичних колекцій (рис. 1.7) є те, що вони формуються на основі фотографії користувача. Після публікації колекції користувач повинен очікувати до 72 годин, поки модератор її підтвердить. Це необхідно для забезпечення відповідності контенту правилам платформи та запобігання публікації небажаних матеріалів.

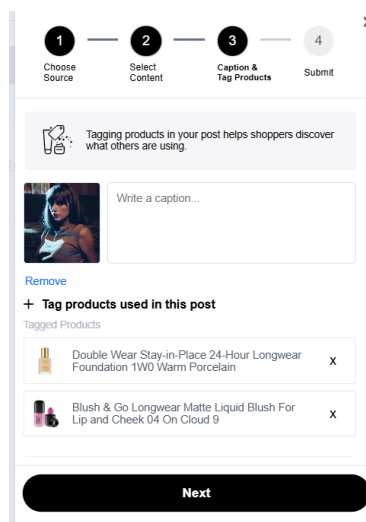


Рис. 1.7. Створення колекції продуктів, що використані на фото

Цей весь функціонал створює відчуття комфортного та об'єднаного ком'юніті, проте платформа має чітку комерційну спрямованість, що призводить до певних обмежень.

Основним недоліком є те, що в системі представлені не всі бренди (рис. 1.8), а лише ті, що представлені в мережі Sephora. Це унеможливорює

створення повноцінної персональної колекції, якщо користувач віддає перевагу іншим брендам. На відміну від Pinterest, тут відсутня гнучкість у структуруванні контенту, користувач не може створювати повноцінні власні тематичні розділи для зберігання колекцій продуктів.

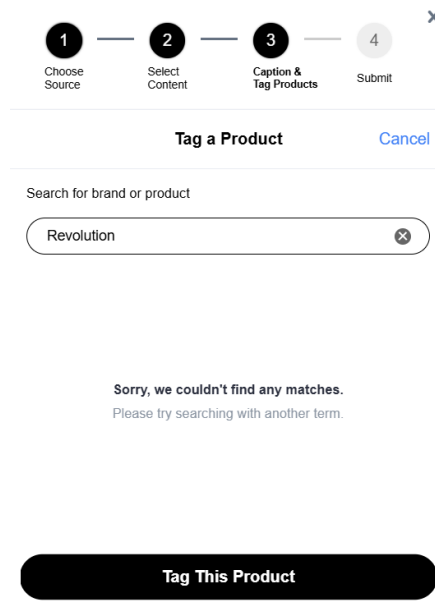


Рис. 1.8. Відсутність необхідних бренду для формування косметичної колекції

Окремим недоліком платформи для українських користувачів є обмеження у зручності її використання, а саме відсутність доступу до європейських версій. У зв'язку з цим користувачі змушені використовувати версію, орієнтовану на ринки США та Канади, що призводить до повільного завантаження сторінок, затримок у відповіді сервера, а також до того, що окремий функціонал може не спрацьовувати з першого разу, що змушує користувачів повторно перезавантажувати сторінку.

Ще один з цікавих аналогів є український сайт SafetyMakeup [4] – його основна мета є аналіз інгредієнтів косметичних продуктів, але він також включає певні додаткові функції, як-от створення косметичних колекцій (створення б'юті-рутини) (рис. 1.9), віш-лісти, порівняння продуктів за їхнім складом, перегляд активних компонентів, з подальшою можливістю їх збереження до себе у профіль, також присутнє додавання продуктів та колекцій до улюблених.

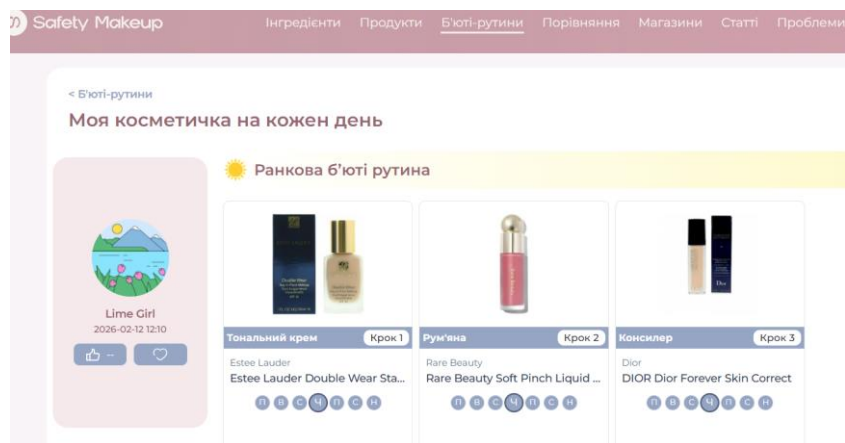


Рис. 1.9. Вигляд створеної косметичної колекції

Основною проблемою у створенні косметичних колекцій (рис. 1.9) є обмежені можливості їх детального налаштування, а саме відсутня можливість обирати конкретний відтінок косметичного продукту. Крім того, жорстке розділення рутин на ранкову та нічну є незручним, оскільки не враховує потреб користувачів у створенні універсальних колекцій, наприклад для продуктів, що використовуються протягом усього дня.

На відміну від платформи Sephora Community, у даному сервісі відсутня повноцінна можливість ведення блогу. Хоча користувачі можуть залишати коментарі під продуктами та косметичними колекціями, така форма взаємодії є обмеженою та не забезпечує повного обміну досвідом між користувачами. Але варто відзначити наявність розділу «Статті» (рис. 1.10), у якому публікується корисний інформаційний контент на актуальні теми, але цей контент створений авторами сайту і як зазначалося вище, не допомагає досягти повноцінної взаємодії між користувачами.

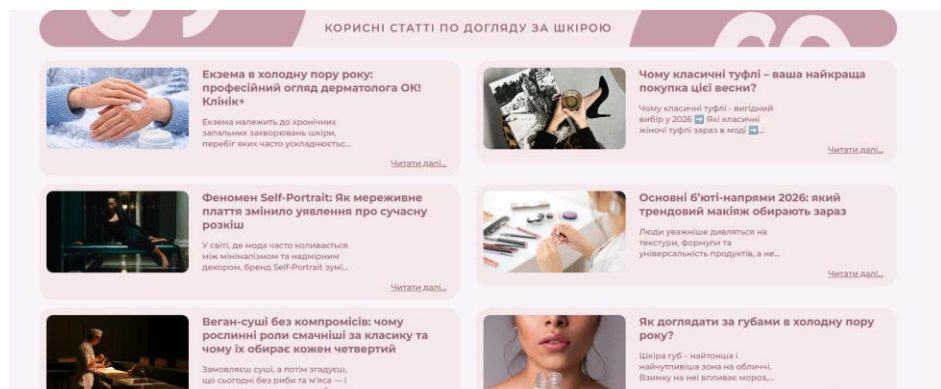


Рис. 1.10. Розділ «Статті» на сайті

Таблиця 1.1

Порівняння функціональних можливостей аналогів

Функціональна можливість	Pinterest	Sephora Community	SafetyMakeup	Розроблювана платформа
Створення власних тематичних колекцій	+ – (обмежено у модифікації)	+ – (тільки до фотографії)	+	+
Додавання продуктів будь-яких брендів	+	–	+	+
Перегляд інформації про продукт використаний у колекції	+ – (можна залишити посилання на магазин)	+	+ – (недоступний перегляд відгуків продукту)	+
Можливість ведення повноцінного особистого блогу	–	+	–	+
Створення різних тематичних розділів для організації	+	–	–	+
Зручність у використанні програмного продукту	+	– (дуже довго відповідає на запити)	+	+
Створення тематичних груп для обговорення конкретної теми	–	+	–	–
Аналіз складу косметичного продукту	–	–	+ (власна база косметичних інгредієнтів)	+ (використання ІІІ)

Порівняльний аналіз функціональних можливостей існуючих аналогів (табл. 1.1) демонструє, що кожна з проаналізованих платформ не повністю задовольняє користувацькі потреби у тій чи іншій мірі. Тому виникає необхідність у розробці платформи, яка об'єднуватиме переваги проаналізованих рішень та усуватиме виявлені недоліки, для забезпечення зручного створення, організації та керування персональними косметичними колекціями з можливістю ведення блогу.

1.3 Вибір архітектури для вебплатформи створення косметичних колекцій та ведення блогу

Для початку розробки проєкту, після визначення вимог, наведених вище, необхідно обрати архітектурний підхід, який буде використано під час створення програмного забезпечення.

Оскільки проєкт реалізується у вигляді вебзастосунку, доцільним рішенням є використання клієнт-серверної архітектури. Основна особливість якої полягає у розподілі функціональності між клієнтською частиною, яка відповідає за взаємодію з користувачем, та серверною частиною, що забезпечує обробку запитів, зберігання даних і керування ресурсами системизш.

Взаємодія між клієнтом і сервером буде здійснюватися за допомогою REST API – архітектурного стилю, що використовує HTTP-методи для обміну даними [5].

Окрім цього, необхідно визначити архітектуру серверної частини застосунку. У межах проєктування було розглянуто два підходи: монолітну та мікросервісну архітектуру.

Мікросервісна архітектура – це архітектурний стиль, за якого програмний застосунок складається з набору невеликих, незалежних сервісів. Кожен сервіс відповідає за окрему бізнес-функцію та взаємодіє з іншими сервісами через мережу за допомогою чітко визначених інтерфейсів. Однією з ключових переваг такого підходу є підвищена відмовостійкість системи, тобто, у разі недоступності одного сервісу інші компоненти можуть продовжувати роботу. Крім того, мікросервіси дозволяють незалежно масштабувати окремі частини системи та використовувати різні технології для різних сервісів.

Важливим недоліком це те що у випадку неправильної організації комунікації між сервісами може виникати надмірна кількість зв'язків, що призводить до складної структури, яку іноді умовно називають «distributed spaghetti». Саме тому мікросервісний підхід потребує ретельного проєктування, централізованого управління та зрілої інфраструктури [6].

Монолітна архітектура, у свою чергу, передбачає створення застосунку у вигляді єдиного цілісного програмного модуля, «моноліту», в межах якого всі функціональні компоненти виконуються в одному процесі. Такий підхід спрощує ранні етапи розробки та тестування, а також зменшує загальну складність архітектури на старті проєкту. Крім того, він є зручним рішенням для швидкого запуску нового програмного продукту з мінімальними витратами на налаштування інфраструктури. У той же час, зі зростанням системи монолітна архітектура може ускладнюватися в підтримці, оскільки всі компоненти щільно пов'язані між собою. Це може призводити до зниження гнучкості системи та ускладнення внесення змін без впливу на інші частини застосунку [6].

Тому після аналізу було прийнято рішення обрати монолітну архітектуру, оскільки мікросервісний підхід є доцільним переважно для великих систем зі значною кількістю незалежних компонентів та складною інфраструктурою. Для проєкту поточного масштабу монолітна архітектура забезпечує простішу реалізацію, зручніше тестування та швидший процес розробки, а також дозволяє уникнути надмірної складності, характерної для розподілених систем. Також слід зазначити, що правильно спроектована монолітна архітектура може бути не менш ефективною порівняно з іншими архітектурними підходами. Навіть найпопулярніші архітектурні рішення можуть призводити до проблем у разі їх неправильної організації або невідповідного застосування в межах програмної системи.

Завдяки обраному архітектурному стилю було прийнято рішення використати підхід Onion Architecture (рис. 1.11) [7]. Він складається з декількох шарів, що взаємодіють між собою та спрямовані до ядра системи, яким є домен. Кожен шар відповідає за окремий рівень логіки та виконує визначену роль у структурі застосунку.

На рис. 1.11 зображено основні шари, що використовуються при побудові системи. За потреби архітектура може бути деталізована шляхом введення додаткових рівнів абстракції або застосування суміжних архітектурних підходів, наприклад, Domain Driven Design. Однак у межах даної роботи розглядаються

чотири основні шари: Domain, Application, Infrastructure та API, кожен з яких виконує свою окрему функцію в загальній архітектурі системи.

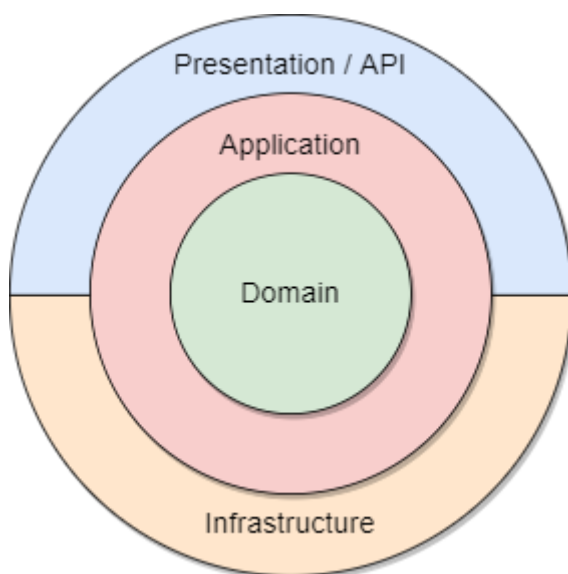


Рис. 1.11. Вигляд шарів у Onion Architecture

Перший шар (Domain) є ядром системи. Він є незалежним від інших шарів і містить основні та бізнес-правила. Наступний шар (Application) відповідає за реалізацію бізнес-логіки та координує взаємодію між доменними сутностями та зовнішніми компонентами системи такими як Infrastructure та API. Зовнішній шар (Infrastructure) відповідає за взаємодію із зовнішніми системами та ресурсами. У його межах реалізовано доступ до бази даних, роботу з ORM, файловими сховищами, а також інтеграцію із зовнішніми сервісами. Саме цей шар забезпечує збереження та отримання даних, реалізуючи технічні деталі доступу до інформації, які не повинні бути частиною бізнес-логіки. Та останній шар API або Presentation layer є зовнішнім рівнем системи та відповідає за взаємодію користувача з програмним продуктом. Він реалізує обробку HTTP-запитів, маршрутизацію, контролери та передачу даних між клієнтською частиною та внутрішніми шарами системи. Основною задачею цього шару є надання зручного та стандартизованого інтерфейсу для доступу до функціональності застосунку.

Для організації коду на клієнті було обрано підхід розбиття на функціональні модулі. Це дозволило згрупувати логіку, компоненти та стилі навколо окремих сутностей, таких як «Колекції» або «Блог». Такий підхід

забезпечує чітке розмежування функціональності системи та зменшує взаємозалежність між її окремими частинами.

1.4 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Для розробки даної платформи обрано сучасний стек вебтехнологій та середовищ для розробки систем, які характеризуються гнучкістю, масштабованістю та активною підтримкою спільноти, що є важливими аспектами під час створення сучасного програмного забезпечення [8].

Для реалізації клієнтської частини застосунку обрано бібліотеку React.js [9], яка забезпечує швидке та динамічне оновлення інтерфейсу завдяки використанню Virtual DOM. Основною особливістю цього підходу є оновлення реального DOM лише в тих місцях, де це необхідно, що позитивно впливає на продуктивність застосунку. Крім того, React.js підтримує компонентний підхід до розробки, що спрощує масштабування системи та повторне використання коду. У процесі вибору технології клієнтської частини також було розглянуто фреймворк Angular [10], який орієнтований на розробку великих корпоративних застосунків зі складною архітектурою та жорстко визначеною структурою. Angular надає широкий набір вбудованих інструментів, проте його використання характеризується вищим порогом входження, що є менш доцільним для проєктів середнього масштабу. React.js, на відміну від Angular, має більш гнучку архітектуру та нижчий поріг входження, особливо для розробників із базовими знаннями JavaScript. Бібліотека не нав'язує жорсткої структури проєкту, що дозволяє обирати додаткові інструменти та бібліотеки відповідно до потреб застосунку.

Мову програмування для клієнтської частини обрано TypeScript – це строго типізована надобудована JavaScript, яка додає статичну типізацію та розширені можливості для роботи з великими проєктами [11]. Завдяки цьому підвищується якість коду, зменшується кількість потенційних помилок на етапі розробки та покращується читабельність та підтримуваність коду.

Серверну частину платформи реалізовано за допомогою фреймворку ASP.NET Core Web API, який вирізняється високою продуктивністю, суворою типізацією та надійними механізмами безпеки, що робить його доцільним вибором для створення масштабованих вебсервісів [12, 13]. Фреймворк підтримує побудову REST API, що забезпечує стандартизовану взаємодію між клієнтською та серверною частинами системи.

Також, ASP.NET Core має вбудовану підтримку DI, що сприяє побудові слабозв'язаних і легко тестованих компонентів. Архітектура фреймворку оптимізована для високої продуктивності та асинхронної обробки запитів, що дозволяє ефективно працювати з великою кількістю одночасних користувачів. Додатковою перевагою є кросплатформеність, що дозволяє розгортати застосунок на різних операційних системах, включаючи Windows, Linux та macOS.

Для зберігання даних у процесі проєктування було розглянуто дві системи керування базами даних, а саме PostgreSQL та SQL Server.

SQL Server є потужною системою керування базами даних із глибокою інтеграцією в екосистему Microsoft, що спрощує взаємодію з .NET-застосунками та інструментами Visual Studio [14]. Також вона надає широкий набір вбудованих інструментів для аналітики, резервного копіювання та адміністрування даних. Однак основними обмеженнями є ліцензійна модель та менша гнучкість у середовищах, відмінних від Windows. Крім того, використання SQL Server може бути менш вигідним у масштабованих або кросплатформних рішеннях через залежність від комерційної ліцензії.

PostgreSQL, у свою чергу, є повністю безкоштовною системою керування базами даних з відкритим вихідним кодом та кросплатформною підтримкою [15]. Вона забезпечує високу продуктивність, надійність та повну відповідність стандарту SQL. Додатковою перевагою є підтримка розширених типів даних, індексів та можливість створення власних розширень, що робить її більш гнучкою для складних проєктів. PostgreSQL також добре масштабується при роботі з великими обсягами даних і підтримує ефективну роботу в умовах

високого навантаження. Відсутність ліцензійних обмежень і можливість використання в різних екосистемах роблять PostgreSQL більш універсальним рішенням для розробки вебзастосунків.

Для зберігання неструктурованих даних було обрано хмарне сховище Azure Blob Storage [16], його основна перевага це те що він легко інтегрується з .NET застосунками, також він забезпечує функції безпеки, включаючи шифрування під час передачі даних, а також зберігає кілька копій даних, щоб захистити їх від будь-яких непередбачених подій.

Для організації взаємодії серверної частини з базою даних було розглянуто дві технології доступу до даних: Dapper та Entity Framework Core.

Dapper є micro-ORM, який забезпечує швидкий та ефективний доступ до бази даних шляхом виконання SQL-запитів безпосередньо [13]. Його основною перевагою є висока продуктивність і відсутність перетворень з LINQ-query в SQL-query, що надає швидкодію, це робить його доцільним для виконання у складних або високонавантажених запитів. Водночас Dapper має обмежені можливості та не підтримує підхід Code First, що ускладнює керування схемою бази даних на етапах подальшого розвитку застосунку.

Entity Framework Core є повноцінним ORM-рішенням, яке дозволяє працювати з базою даних за допомогою .NET-об'єктів [17], підтримує підхід Code First, який дозволяє реалізувати моделі домену у вигляді POCO-класів, та механізм міграцій [18]. Незважаючи на дещо нижчу продуктивність порівняно з Dapper, використання EF Core забезпечує зручність розробки, що є доцільним для даного проєкту.

Для автоматичного формування описів косметичних продуктів та аналізу їхніх інгредієнтів було використано велику мовну модель GPT-4. На основі наданого складу та характеристик продукту модель генерує узагальнену інформацію щодо його призначення, потенційних переваг та можливих ризиків. Інтеграція функціоналу штучного інтелекту в систему використано пакет Semantic Kernel, який забезпечує взаємодію між застосунком та мовною моделлю, керування запитам до неї та обробку отриманих результатів [19].

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

Середовище розробки серверної частини обрано Visual Studio – це інтегроване середовище розробки (IDE) від компанії Microsoft, доступне для платформ Windows та macOS, яке надає широкий набір інструментів для розробки .NET-застосунків. Основними перевагами є глибока інтеграція з платформою .NET, а також вбудовані інструменти для тестування.

Для розробки клієнтської частини було використано редактор коду Visual Studio Code є безкоштовним та інструментом, що працює на операційних системах Windows, macOS та Linux. Його основними перевагами пропонує чудову підтримку широкого спектру мов програмування, включаючи TypeScript, велика кількість розширень та можливість налаштування середовища під потреби користувача.

Висновки до першого розділу

У першому розділі випускної кваліфікаційної роботи було проаналізовано основні проблеми, з якими стикаються користувачі під час взаємодії з інформацією про косметичні продукти у сучасному цифровому середовищі. Було виявлено відсутність зручних інструментів для організації косметичних засобів, недостатньо сучасний та інтуїтивний дизайн, а також обмежені можливості соціальної взаємодії між користувачами.

Окрім цього було встановлено, що існуючі рішення часто не забезпечують достатнього рівня персоналізації та не дозволяють користувачам ефективно організовувати власні косметичні колекції. Це ускладнює процес управління інформацією та знижує загальну зручність використання подібних платформ.

Отримані результати аналізу стали основою для визначення ключових напрямів її розробки.

Також у ході дослідження було розглянуто існуючі аналоги програмних рішень, зокрема платформи Pinterest, Sephora Community та SafetyMakeup. Аналіз продемонстрував, що дані платформи частково задовольняють потреби користувачів у створенні косметичних колекцій і обміні досвідом. Pinterest демонструє потужні функції для візуального натхнення та збереження ідей, проте йому бракує можливості соціальних взаємодій. Sephora Community, у свою

чергу, надає розширені інструменти для комунікації та обміну відгуками, але має комерційну спрямованість і тому має обмежений асортимент брендів, що зменшує можливості для формування повноцінної персональної косметичної колекції. Платформа SafetyMakeup дозволяє користувачам створювати косметичні колекції та надає можливість аналізу інгредієнтів у косметичних продуктах. Але має обмежені можливості щодо детального налаштування колекцій, а також недостатньо розвинену складову соціальної взаємодії між користувачами.

Також було визначено архітектурний підхід для майбутньої побудови застосунку. Обрано клієнт-серверну архітектуру, що забезпечує чітке розмежування відповідальностей між клієнтською та серверною частинами системи. Для реалізації серверної логіки обрано монолітний стиль архітектури, який є доцільним для застосунків середнього масштабу та дозволяє спростити розробку, тестування й підтримку системи на початкових етапах, а для клієнтської частини було обрано підхід на розділення на функціональні модулі.

Окрім цього, було проаналізовано сучасні технології, які широко використовуються у розробці вебзастосунків та є актуальними в реальних проєктах. Основна увага приділялася інструментам, що забезпечують стабільність, масштабованість та зручність підтримки програмного продукту.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБПЛАТФОРМИ ДЛЯ СТВОРЕННЯ КОСМЕТИЧНИХ КОЛЕКЦІЙ ТА ВЕДЕННЯ БЛОГУ

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

Вебплатформа для створення косметичних колекцій та ведення блогу призначена для надання користувачам зручного інструменту для організації власних косметичних засобів і обміну досвідом. Відповідно до цього, до платформи висувається низка вимог, що визначають її основну функціональність (табл. 2.1).

Таблиця 2.1

Вимоги користувачів системи

Користувачі	
Колекції	Мати можливість створювати та наповнювати косметичні колекції різними продуктами.
	Мати можливість створення розділів та додавання до них різних косметичних колекцій.
	Мати можливість додавання інших користувачів колекцію у розділи в секції «Вибране».
	Мати можливість приховувати колекції за бажанням.
Дописи	Мати можливість створювати дописи на вебплатформі.
	Мати можливість окрім текстової частини, додавати і інші елементи колекції, продукти, тощо.
	Мати можливість приховування дописів.
Продукти	Мати можливість переглядати інформацію про продукт.

продовж. табл. 2.1

Продукти	Мати можливість проаналізувати склад продукту та отримати підсумок про продукт.
	Мати можливість додавання запиту на додавання продуктів чи додаткового варіанту для нього.
Адміністратор	Мати можливість обробки запитів на додавання.
	Мати можливість додавання продуктів та варіантів для них.

Узагальнюючи отримані результати, було сформовано функціональні вимоги, які визначають ключові сценарії роботи системи та описують основні можливості вебплатформи (табл. 2.2).

Таблиця 2.2

Функціональні вимоги

Користувач	Вебплатформа повинна на давати можливість реєстрації та авторизації на платформі.
	Можливість користувачу змінювати інформацію, додавати теги до профілю або змінювати пароль чи електронну пошту та видалення акаунту з платформи.
Колекції	Платформа повинна забезпечувати користувачам можливість створення колекції її наповнення зміни інформації про неї та видалення.
	Можливість іншим користувачам додавати колекцію іншого користувача у розділ «Вибране» з можливістю створення розділів у ньому.

Колекції	Фільтрування продукції у колекції за різними характеристиками, а також пошук продуктів у ній.
	Створення та збереження колекцій у різних розділах.
	Переміщення колекцій по розділам відповідно до секції (вибране, особисті).
Дописи	Створення, видалення та редагування дописів на платформі.
	Можливість іншим користувачам реагувати, коментувати та зберігати пости у «Переглянути пізніше».
	Можливість редактору додавати колекції, продуктів, згадок інших користувачі у пости та продуктів до посту.
Продукти	Перегляд продуктів у каталозі, з можливістю їхньої фільтрації та пошуку їх.
	Переглядати інформацію про продукт та можливість оцінювання та коментування продуктів.
	Можливість розібрати склад продукту та отримати висновок на основі інгредієнтів та категорії.
Пошук	Пошук різних об'єктів за їхніми характеристиками.
Адміністратор	Обробка запитів на додавання продуктів та варіантів, з можливістю відхилити запит, якщо він є невалідним.
	Додавання продуктів та варіантів до продукту у системі.

Таким чином, визначені функціональні вимоги формують основу основні можливості вебплатформи. Окрім функціональних аспектів, важливу роль у забезпеченні якості програмного продукту відіграють нефункціональні вимоги.

Основні нефункціональні вимоги:

- час, необхідний для навчання звичайних користувачів – 1 година, а для навчання просунутих користувачів – 30 хвилин;
- час відгуку для типових задач – не більше 5 секунд, для складних завдань не більше 10 секунд;
- максимальна норма помилок та дефектів в роботі системи – 3 помилки на десять тисяч рядків коду;
- - всі паролі користувачів від профілів в базі даних повинні зберігатися у зашифрованому вигляді;
- інтерфейс вебплатформи, повинен бути адаптивний під різні пристрої та зрозумілий у використанні для будь якого користувача.

Визначено три основні ролі (актори), що взаємодіють із вебплатформою, а саме незареєстрований користувач (гість), зареєстрований користувач та адміністратор. Незареєстрований і зареєстрований користувачі є основними акторами, які безпосередньо взаємодіють із функціоналом платформи. Адміністратор у свою чергу займається обробкою запитів користувачів на додавання продуктів, а також наповнення вебплатформи. Детальний розбір виявлених акторів наведено в таблиці 2.1.

Таблиця 2.3

Виявлення акторів

Актор	Короткий опис
Адміністратор	Адміністратор системи обробляє запити користувачів на додавання продуктів чи варіантів, додає їх у систему.

Актор	Короткий опис
Зареєстрований користувач	Створює, персональні колекції, переглядає чужі колекції та додає їх у «Вибране», створює нові пости, підписується на інших, секції для розміщення колекцій, додає коментарі та вподобайки.
Гість	Має обмежений доступ, може переглядати продукти публічні колекції, пости без можливості збереження, реакцій або коментування.

Діаграми використання вебплатформи, що демонструють, які дії можуть виконувати користувачі з платформою.

На рис. 2.1 зображено варіанти використання для входу та реєстрації в систему, також користувач може відновити пароль в разі втрати.

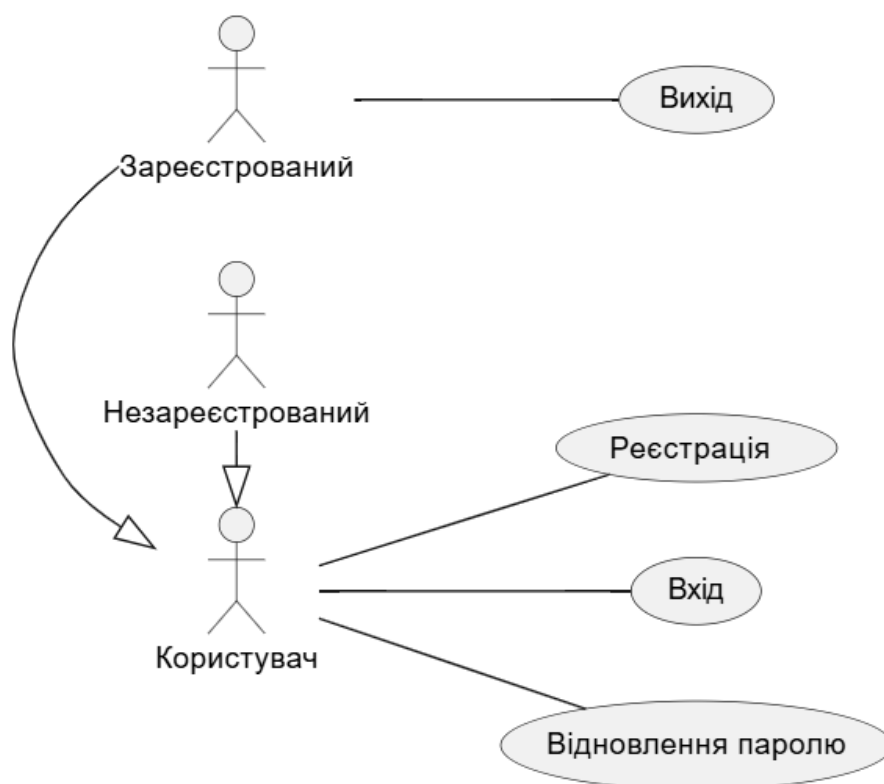


Рис. 2.1. Діаграма використання «Робота з входом та реєстрація»

На рис. А.1 зображено наступну діаграма використання, яка показує варіанти взаємодії користувача з колекціями у системі. Інша діаграма

використання (рис. 2.2) демонструє роботу користувача з постами, де зображено можливості створення, видалення, тощо.

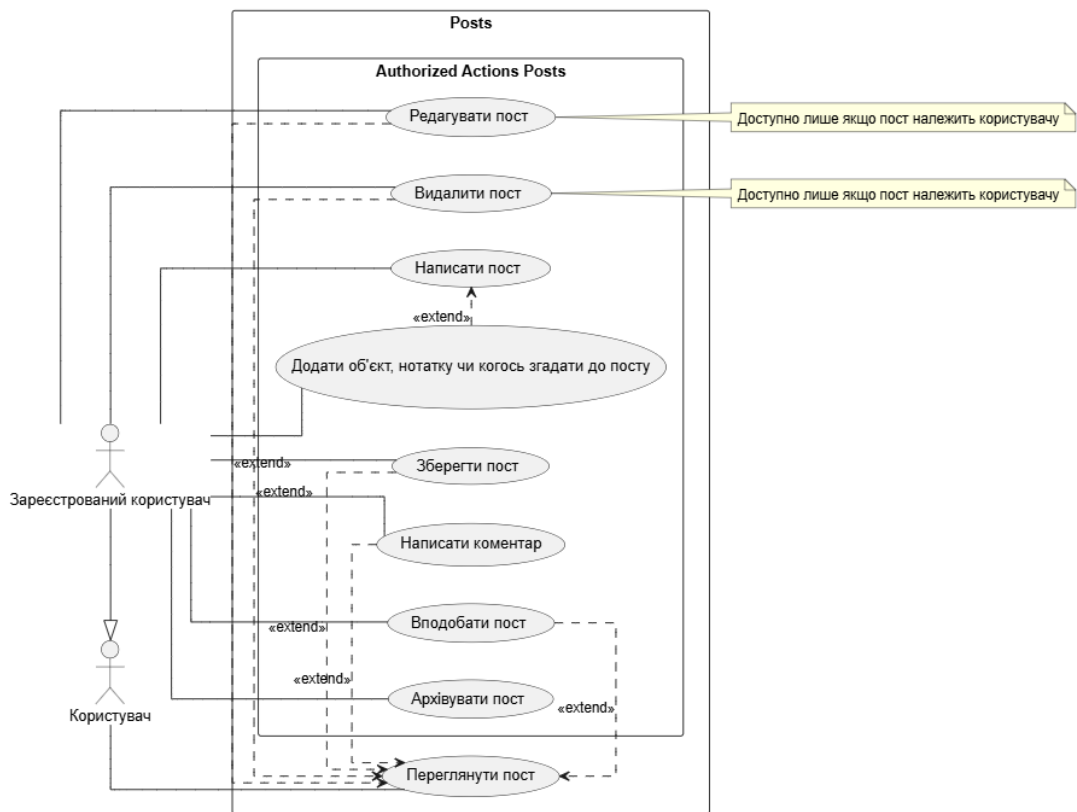


Рис. 2.2. Діаграма використання «Робота з постами»

Інша діаграма використання (рис. 2.3), демонструє основні сценарії взаємодії користувача з продуктами на платформі, в якій можна побачити що користувач може оцінювати продукт, додати продукт у колекцію, перевірити склад, тощо.

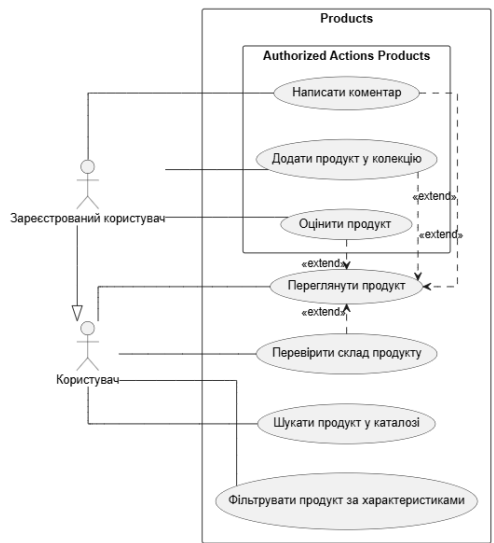


Рис. 2.3. Діаграма використання «Робота з продуктами»

Наступна діаграма (рис. 2.4) використання демонструє основні можливості користувача роботи зі своїм профілем. Можна побачити, що користувач може виконувати основні дії з профілем, а саме налаштовувати інформацію про нього, керування приватністю, тощо.

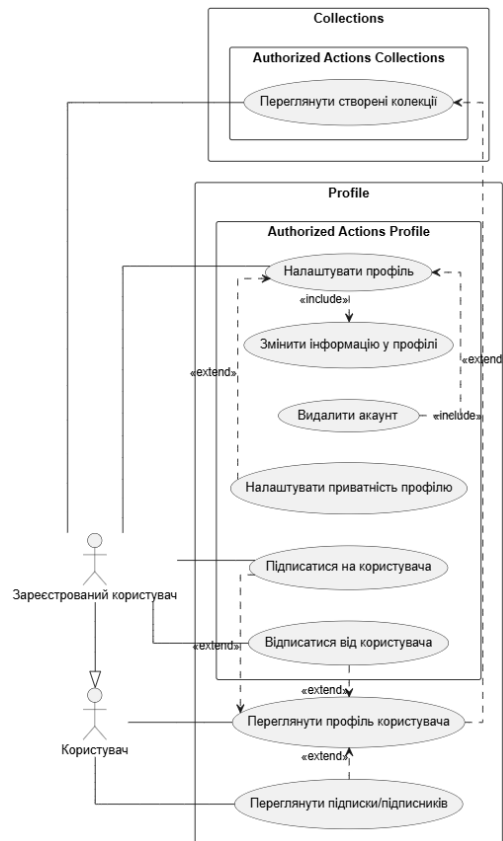


Рис. 2.4. Діаграма використання «Профіль користувача»

Остання діаграма варіантів використання, зображена на рис. 2.5, демонструє основні дії адміністратора у системі. Зокрема, до його функціональних обов'язків належить обробка запитів на додавання нових косметичних продуктів, що передбачає перевірку поданих даних та їх подальше внесення до системи.

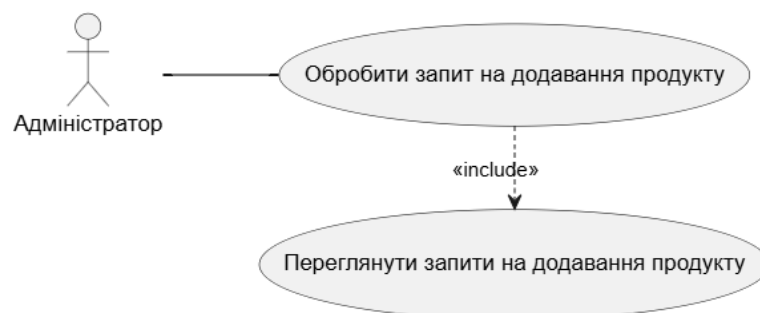


Рис. 2.5. Діаграма використання «Можливості адміністратора»

Опис деяких сценаріїв використання наведено в таблицях 2.4-2.6. Перший прецедент описує кроки для переміщення розділу користувача у інший, що дозволяє гнучко організовувати розділи для косметичних колекцій.

Таблиця 2.4

Опис прецеденту «Переміщення розділу»

Компоненти	Опис
Назва	Переміщення розділу у інший батьківський розділ.
Мета	Надання користувачу можливості змінювати структуру власних розділів у системі для більш зручної організації контенту.
Основний актор	Зареєстрований користувач.
Передумови	1. Користувач авторизований у системі. 2. У користувача існують створені розділи. 3. Обраний розділ може бути переміщений до іншого батьківського розділу (дотримана ієрархічна структура системи).
Умови успішного завершення	Розділ успішно переміщено до нового батьківського розділу, а оновлена структура збережена в системі.
Умови невдалого завершення	Переміщення не виконано у разі переміщення розділу до своїх нащадків, місце нового розташування розділу вказує на самого себе, не є власником.
Основний сценарій успіху	Користувач обирає розділ, який необхідно перемістити, після чого вибирає новий розділ зі списку доступних. Система виконує перевірки, після чого оновлює зв'язки між розділами та зберігає зміни.

Наступний прецедент описує процес редагування існуючого посту, який є важливою складовою блогової частини системи. Він демонструє можливості користувача щодо оновлення власного контенту та підтримання його актуальності. Важливою особливістю даної можливості є те що у користувача буде створюватися окрема копія оригінального посту, де він зможе редагувати допис без впливу на оригінальну.

Опис прецеденту «Редагування посту»

Компоненти	Опис
Назва	Редагування існуючого посту.
Мета	Можливість змінювати вміст уже створеного посту для актуалізації або уточнення інформації.
Основний актор	Зареєстрований користувач.
Передумови	1. Бути авторизованим у системі. 2. Пост у статусі «Опубліковано». 3. Є автором даного допису.
Умови успішного завершення	Зміни у пості успішно збережені та відображаються іншим користувачам платформи.
Умови невдалого завершення	Редагування не виконано через відсутність можливостей до його редагування.
Основний сценарій успіху	Користувач відкриває власний пост, обирає редагувати, вносить необхідні зміни у вміст та підтверджує редагування. Система перевіряє дані та зберігає оновлений варіант посту.

Далі розглядається прецедент створення косметичної колекції, який є одним із ключових функціональних сценаріїв платформи. Він описує процес формування персональних колекцій користувача та їх подальшої організації у межах системи.

Опис прецеденту «Створення косметичної колекції»

Компоненти	Опис
Назва	Створення косметичної колекції.
Мета	Створення персональної косметичної колекції для організації косметичних продуктів.
Основний актор	Зареєстрований користувач.
Передумови	1. Користувач авторизований у системі. 2. Обраний розділ, де буде розміщена колекція, належить користувачу, не є розділом у «Вибране».

Компоненти	Опис
Умови успішного завершення	Нова косметична колекція успішно створена та збережена у профілі користувача.
Умови невдалого завершення	Створення колекції не відбулося через некоректні дані, користувач не є власником розділу чи розділ розміщений у «Вибране».
Основний сценарій успіху	Користувач ініціює створення нової колекції, вводить її назву, опис та видимість колекції. Система виконує перевірку даних і зберігає нову колекцію.

Завдяки проведеному аналізу та обраній архітектурі було побудовано діаграму класів серверної частини програмного продукту.

Шар Domain (рис. Б.2) містить основні сутності предметної області, а також інтерфейси репозиторіїв, які визначають набір методів для роботи з даними та підлягають реалізації в інших шарах системи. Крім того, у цьому шарі розміщено об'єкти для реалізації пагінації, патерн Result, опис якого наведено нижче, а також перелічення та константи.

Шар Application (рис. Б.1) відповідає за реалізацію бізнес-логіки та обробку запитів, що надходять із API. Він координує виконання операцій, взаємодію між доменними об'єктами та забезпечує виконання основних сценаріїв використання системи.

Шар Infrastructure (рис. Б.3) забезпечує взаємодію із зовнішніми ресурсами, такими як база даних, фонові процеси (background worker), файлове сховище, робота зі штучним інтелектом та конфігураційні параметри системи. Він реалізує технічні деталі доступу до даних та зовнішніх сервісів.

Шар API (рис. Б.4) виступає точкою входу до системи та забезпечує взаємодію клієнтських застосунків із серверною частиною. Він обробляє HTTP-запити, передає їх до відповідних сервісів шару Application та повертає результати у вигляді відповідей клієнту.

Також під час проєктування платформи також було використано низку шаблонів проєктування. Зокрема, патерн Стратегія [20], який реалізує різні типи взаємодії з об'єктами. Аналіз діаграми класів (рис. 2.7) показує наявність двох типів інтерфейсів. Перший відповідає за роботу з об'єктами, що не мають власника, тоді як другий розширює його функціональність, додаючи можливість роботи з автором об'єкта. Надалі ці інтерфейси використовуються як основа для абстрактних класів, у яких реалізовано спільну логіку взаємодії з об'єктами. Водночас частина методів залишається абстрактною та повинна бути визначена в похідних класах відповідно до їхнього призначення [21]. Такий підхід забезпечує повторне використання спільного функціоналу та дозволяє гнучко адаптувати поведінку для різних типів об'єктів.

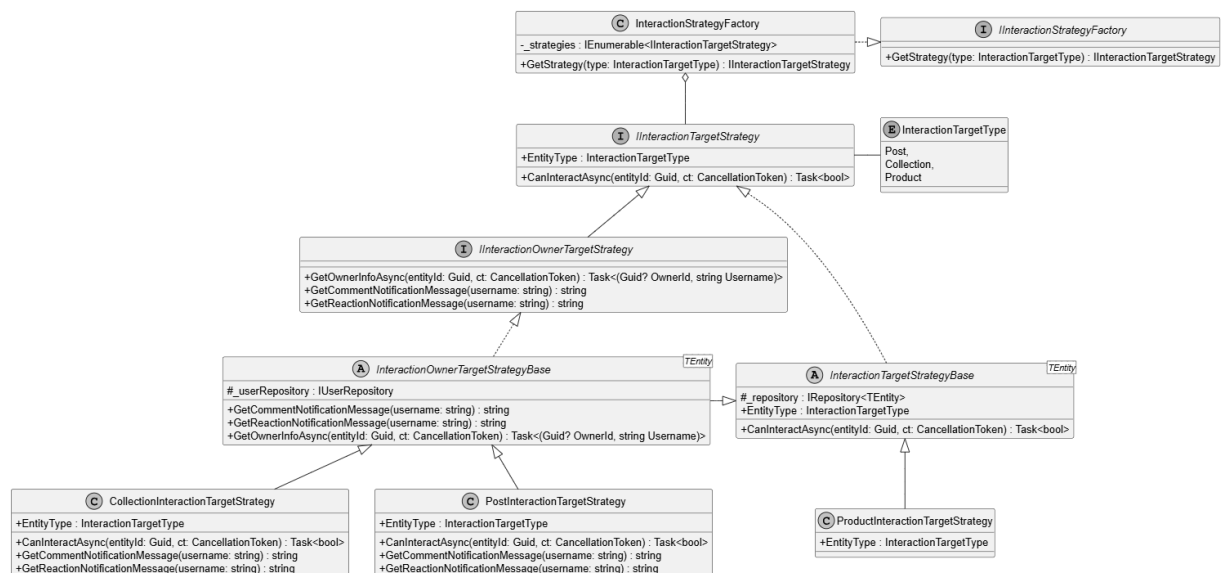


Рис. 2.6. Діаграма класів, що реалізує патерн Стратегія

Наступним використаним шаблоном є Будівельник [20] (рис. Б.5), основною задачею якого є формування логіки перевірки доступу користувача до взаємодії з об'єктами системи. Його застосування зумовлене тим, що різні типи об'єктів вимагають різного підходу до перевірки доступу. Наприклад, для коментарів і реакцій, які не мають прямого зв'язку з власником через навігаційні властивості, необхідно додатково отримувати інформацію про їх авторів за допомогою механізмів, реалізованих у патерні Стратегії, тоді як для інших об'єктів використовується більш прямий підхід через їхні сутності.

Також у проєкті застосовано патерн Repository, основна ідея якого полягає у відокремленні логіки доступу до даних (зокрема виконання запитів до бази даних) від бізнес-логіки застосунку. Це дозволяє спростити тестування та забезпечити гнучкість у заміні джерел даних

Додатково реалізовано патерн Result, який забезпечує уніфікований підхід до обробки результатів виконання операцій. Він дозволяє уникати використання виключень як основного механізму керування потоком виконання, натомість явно повертаючи результат операції з відповідним статусом успішності або помилки, що підвищує передбачуваність роботи системи.

2.2 Розробка бази даних системи

У процесі проєктування структури бази даних було створено 36 таблиць, діаграма бази даних (рис. В.1). Слід зазначити, що частина таблиць є частиною ASP.NET Identity [22], оскільки для реалізації механізмів реєстрації, авторизації та автентифікації користувачів використовувався даний фреймворк. Він автоматично створює набір таблиць (для зберігання користувачів, ролей, прав доступу та токенів), які інтегруються в загальну структуру бази даних, тому деякі таблиці або поля безпосередньо не використовуються.

Для кращого розуміння структури системи доцільно детальніше розглянути основні сутності предметної області, їх призначення та взаємозв'язки. Опис цих сутностей наведено в таблицях нижче.

Таблиця «Users» (табл. 2.6) є однією з ключових сутностей системи та призначена для зберігання інформації про зареєстрованих користувачів платформи. Вона містить дані профілю користувача, параметри конфіденційності, а також містить поля необхідні для злагодженої роботи ASP.NET Identity.

Таблиця 2.7

Структура таблиці «Users»

Field Name	DataType	PK	FK	Description
ID	uuid	+	–	ID користувача
FirstName	character varying(50)	–	–	Ім'я користувача

Field Name	DataType	PK	FK	Description
LastName	character varying(100)	–	–	Прізвище користувача
RefreshToken	text	–	–	Значення Refresh Token
RefreshTokenExpiresAtUTC	timespan	–	–	Термін дії Refresh Token
ProfileDescription	character varying(500)	–	–	Опис профілю користувача
IsPrivate	boolean	–	–	Чи приватний профіль
UserName	character varying(256)	–	–	Ім'я користувача
NormalizedUserName	character varying(256)	–	–	Нормалізоване ім'я користувача
Email	character varying(256)	–	–	Електронна пошта
NormalizedEmail	character varying(256)	–	–	Нормалізована електронна пошта
PasswordHash	text	–	–	Хеш паролю

Наступна таблиця «Collections» (табл. 2.8), вона відповідає за збереження колекцій користувача. Вона пов'язана з таблицями «Users» зв'язком один-до-багатьох, тобто користувач може декілька колекція, тоді як колекція може мати одного користувача, також за таким самим принципом має зв'язок з таблицею «Sections». Варто зазначити, що також дана таблиця має зв'язок багато-до-багатьох з таблицею «Products», «TagCollections» та «SavedCollections».

Таблиця 2.8

Структура таблиці «Collections»

Field Name	DataType	PK	FK	Description
ID	uuid	+	–	ID колекції
UserId	uuid	–	+	ID користувача, якому належить колекція
SectionId	uuid	–	+	ID секції де розміщена колекція
Title	character varying(300)	–	–	Заголовок колекції
Description	character varying(1000)	–	–	Опис до колекції
IsPrivate	boolean	–	–	Чи приватна колекція
CreatedAt	timestamp	–	–	Коли була створена колекція

Field Name	DataType	PK	FK	Description
EditedAt	timestamp	-	-	Коли була відредагована колекція

Таблиця «Posts» (табл. 2.9) призначена для зберігання інформації про публікації користувачів на платформі. Крім зв'язків з іншими сутностями системи, у ній реалізовано самореферентний зв'язок типу «один-до-одного», за якого запис може посилатися на інший запис цієї ж таблиці. Такий підхід використовується для створення копії опублікованого посту під час його редагування. Це дозволяє зберігати оригінальну версію публікації без змін до моменту підтвердження та збереження нової версії, забезпечуючи цілісність даних.

Таблиця 2.9

Структура таблиці «Posts»

Field Name	DataType	PK	FK	Description
Id	uuid	+	—	ID посту
UserId	uuid	—	+	ID користувача, якому належить пост
Title	character varying(300)	—	—	Заголовок колекції
ContentJson	text	—	—	Наповнення посту, для коректного виведення на клієнті
PostStatus	integer	—	—	Стан посту (Draft, Published, Archived)
ContentText	character varying(2000)	—	—	Наповнення посту в текстовому вигляді
PublishedAt	timestamp	—	—	Коли опубліковано
CoverImageUrl	text	—	—	Зберігання фотографія для попереднього перегляду
ShortDescription	text	—	—	Короткий опис до допису
OriginalPostId	uuid	—	+	ID оригінального посту

Таблиця «Products» (табл. 2.10) призначена для зберігання інформації про косметичні продукти. Окрім стандартних полів, вона містить два атрибути типу jsonb, що використовуються для зберігання вкладених структур даних. У моделі

EF Core ці структури представлені як залежні об'єкти (Owned Entities), які налаштовуються за допомогою конфігурацій OwnsOne. Оскільки такі об'єкти не мають самостійного життєвого циклу та існують лише в межах продукту, їх зберігання у форматі JSON дозволяє уникнути створення додаткових таблиць та спростити структуру бази даних.

Таблиця 2.10

Структура таблиці «Products»

Field Name	DataType	PK	FK	Description
Id	uuid	+	–	ID продукту
Name	text	–	–	Назва продукту
Description	character varying(1000)	–	–	Опису продукту
ProductCategoryId	uuid	–	+	ID категорії, якому належить продукт
IsDiscontinued	boolean	–	–	Чи випускається продукт досі
Ingredients	text	–	–	Список інгредієнтів продукту
BrandId	uuid	–	+	ID бренду
Characteristics	jsonb	–	–	Особливості продукту
AddedAt	timestamp	–	–	Коли було додано продукт
Summary	jsonb	–	–	Висновки про продукт, зроблені ШІ
DeletedAt	timestamp	–	–	Коли видалено продукт
IsDeleted	boolean	–	–	Чи видалений продукт

Таблиця «Comments» призначена для зберігання коментарів користувачів у системі. Особливістю даної сутності є відсутність прямих зв'язків із об'єктами, до яких може бути залишений коментар. Тому «зв'язок» реалізовано через універсальні поля, які містять тип об'єкта та його ідентифікатор (ID). Додатково у таблиці реалізовано самореферентний зв'язок, який забезпечує можливість створення вкладених (деревоподібних) коментарів, де кожен коментар може мати батьківський коментар.

Таблиця 2.11

Структура таблиці «Comments»

Field Name	DataType	PK	FK	Description
Id	uuid	+	–	ID коментарю
UserId	uuid	–	+	ID користувача, якому належить коментар
TargetId	uuid	–	–	ID якому об'єкту належить коментар
ParentId	uuid	–	+	ID коментарю, якому належить коментар (відповідь на коментарі)
Text	character varying(600)	–	–	Текст коментарю
CreatedAt	timestamp	–	–	Час створення

Таблиця «Ingredients» зберігає інформацію про інгредієнти у косметичних засобах та має зв'язок типу один-до-багатьох із таблицею «IngredientName Variations». Даний зв'язок реалізовано для врахування того, що різні виробники можуть використовувати різні назви одного й того самого інгредієнта. Для оптимізації обробки даних та зменшення витрат ресурсів при пошуку, замість дублювання інформації використовується розділення на дві таблиці. Це дозволяє зберігати базовий інгредієнт окремо та пов'язані з ним варіації назв.

Таблиця 2.10

Структура таблиці «Ingredients»

Field Name	DataType	PK	FK	Description
ID	uuid	+	-	ID інгредієнту
DangerLevel	integer	-	-	Рівень небезпеки для людини
Name	text	-	-	Назва інгредієнту
Description	text	-	-	Опис інгредієнту

2.3 Проєктування та реалізація алгоритмів роботи системи

Після визначення архітектури програмного продукту та структури бази даних наступним етапом є проєктування та реалізація алгоритмів роботи системи. У межах даного розділу було виділено основні бізнес-алгоритми, що забезпечують функціонування платформи.

До них належать алгоритми редагування та публікації постів, а також механізм оновлення інформації про інгредієнти косметичних продуктів у системі.

Розглянемо процес редагування посту (рис. 2.7). Коли користувач у клієнтській частині системи обирає функцію редагування, на сервер надсилається відповідний запит. На серверній стороні виконується перевірка прав доступу (визначається, чи є користувач автором даного посту) та чи існує заданий пост у базі даних. У випадку, якщо користувач не є власником, система повертає помилку про заборону виконання операції. У разі успішної перевірки система отримує поточний стан посту та перевіряє його статус. Якщо пост уже знаходиться у стані чернетки, користувачу повертається ідентифікатор існуючої чернетки без створення нової версії.

Далі перевіряється наявність активної чернетки для даного посту. Якщо чернетка вже існує для обраного посту то, система повертає її ідентифікатор. У випадку її відсутності створюється копія поточного посту, синхронізуються пов'язані сутності (зокрема нотатки та зображення), оновлюється контент чернетки, після чого вона зберігається у базі даних.

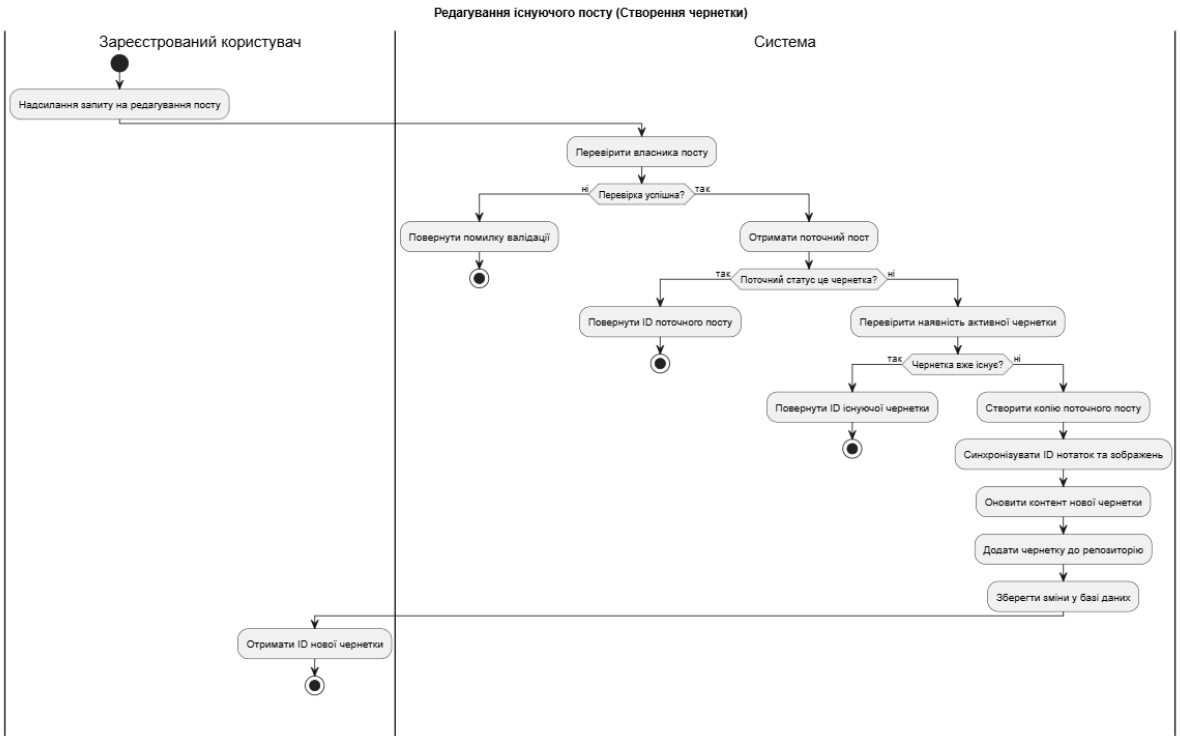


Рис. 2.7. Діаграма активності «Редагування існуючого посту»

Наступним етапом варто розглянути алгоритм, що виконується, коли користувач, хоче або опублікувати пост (рис. Г.1). Користувач заповнює форму попереднього перегляду посту та ініціює запит на його публікацію. Далі запит передається на сервер, де виконується комплексна перевірка коректності та можливості виконання операції.

На першому етапі система перевіряє, чи є користувач автором відповідного посту. У разі невідповідності прав доступу операція завершується з помилкою. Якщо перевірка успішна, система переходить до обробки даних посту, які надходять у форматі JSON. Далі виконується перевірка основних параметрів посту, зокрема заголовка та текстового вмісту.

Якщо поле короткого опису не заповнене, але основний текст посту наявний, система автоматично генерує або оновлює короткий опис на основі довжини контенту. Після цього виконується обробка тегів, які додаються відповідно до обраних даних. У разі помилки обробки тегів публікація не виконується.

Наступним етапом є перевірка можливості переходу у статус «Опубліковано». Якщо перехід у даний стан заборонений операція завершується з помилкою. У разі успішного проходження всіх перевірок пост переходить у стан публікації та зберігається в системі. Додатково, якщо пост містить згадки інших користувачів, система ініціює процес надсилання відповідних сповіщень.

Далі, розглянемо діаграму послідовності (рис. Г.2), де зображено процес оновлення списку інгредієнтів у системі, де відбувається обробка кожного інгредієнта зі списку, а також формування висновку за інгредієнтами та категорією продукту на основі отриманих даних.

Процес ініціюється адміністратором шляхом надсилання запиту на оновлення списку інгредієнтів. Контролер передає дані до сервісного шару, де виконується перевірка наявності відповідного продукту та оновлення його інгредієнтів. Після успішного оновлення даних система ініціалізує фонове завдання через планувальник задач, для продовження роботи системи без блокування основного потоку виконання.

Фоновий процес отримує ідентифікатор продукту та користувача та передає ці дані у відповідний сервіс. У випадку відсутності інгредієнтів до продукту користувачу надсилається відповідне сповіщення про помилку. Якщо інгредієнти присутні, запускається процес їх аналізу.

На цьому етапі система виконує ідентифікацію інгредієнтів. Спочатку здійснюється пошук відомих назв інгредієнтів у базі даних за допомогою відповідних репозиторіїв, використовуючи дві пов'язані таблиці, описані у підрозділі 2.2.

Після цього система аналізує список інгредієнтів та визначає невідомі значення, які відсутні у базі даних. У випадку виявлення таких елементів виконується звернення до великої мовної моделі через Semantic Kernel, який виступає як проміжний шар для формування запиту та обробки відповіді. Результат роботи моделі повертається у форматі JSON, що містить структуровані характеристики інгредієнтів.

Далі отримані дані десеріалізуються та зберігаються у базі даних, після чого формується повний набір інгредієнтів продукту. На основі цього набору виконується генерація узагальненого опису продукту з урахуванням його категорії та складу, також із використанням великої мовної моделі.

Окремо варто зазначити, що весь процес запускається у фоновому режимі за допомогою Quartz Scheduler. Це дозволяє виконувати обробку інгредієнтів асинхронно, без блокування основного потоку користувацьких запитів.

2.4 Реалізація вебплатформи для створення косметиних колекцій та ведення блогу

Для реалізації власного редактору у системі використовується фреймворк Slate JS для створення rich-text редакторів [23]. Він надає можливість повного контролю над структурою документа, поведінкою елементів та їхнім форматуванням, що дозволяє розробляти кастомізовані редактори під конкретні потреби проєкту. В даному випадку він нам потрібен для того, щоб додавати у дописи різні об'єкти (колекції, продукти, нотатки, тощо).

Основна особливість архітектури Slate JS полягає в тому, що вона базується на поділі даних на два основні типи об'єктів: Element та Leaf. Головна різниця між ними полягає у їхній структурі та можливостях взаємодії. Об'єкт типу Element являє собою блокову структуру, яка в межах розробленої платформи використовується для вбудовування складних сутностей, наприклад, списки продуктів чи косметичні колекції.

Лістинг коду блочного елемента, згадування користувача у пості:

```
<span
  {...attributes}
  contentEditable={false}
  className="inline-flex items-center rounded bg-brand-pink/20 px-1.5 py-0.5 text-sm font-medium text-brand-pink mx-1 shadow-sm"
>
  <Link
    to={` /profile/${element.userId}`}
    className="inline-flex justify-center items-center gap-1"
  >
    @{{element.username}}
    <Avatar
      src={{element.avatarUrl}}
      alt={{element.username || "Невідомий"}}
      className="border object-contain! border-surface-blue"
      size={{18}}
    />
  </Link>

  {{children}}
</span>
```

Визначає, цей елемент є блочним його властивості а саме contentEditable={false} і далі потрібно цей елемент зареєструвати у відповідному створеному самостійно плагіні з використанням функції isVoid(element).

Користувач не може змінювати внутрішню текстову структуру таких блоків безпосередньо під час набору тексту, оскільки вони функціонують як цілісні, самостійні компоненти. Натомість об'єкт типу Leaf працює на рядковому рівні і представляє собою більш гнучкий, «плаваючий» елемент документа. У реалізованому редакторі властивості типу Leaf використовуються для створення користувацьких нотаток та виділення тексту.

Після внесення змін користувачем Slate.js формує JSON-дерево документа, у якому зберігається структура тексту, його форматування та вбудовані

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

елементи. Саме це дерево використовується під час створення та редагування чернеток постів, оскільки система зберігає поточний стан документа у базі даних і може відновити його для подальшого редагування.

Під час редагування опублікованого посту створюється його окрема чернетка. Такий підхід дозволяє вносити зміни без впливу на вже опубліковану версію контенту. При цьому недостатньо створити лише копію самого посту, оскільки він пов'язаний з іншими сутностями, зокрема тегами, нотатками та зображеннями. Наприклад, зображення на відміну від більшості даних, зберігаються не безпосередньо у базі даних, а у хмарному сховищі, тому система повинна відстежувати, чи використовується конкретне зображення у поточній версії документа. Тому під час створення чернетки виконується створення дублікатів пов'язаних об'єктів та оновлення посилань на них у JSON-дереві редактора.

Лістинг коду для зміни структури JSON-дерева:

```
if (node == null) return;

if (node is JsonObject obj)
{
    if (obj.ContainsKey(idKey) && obj[idKey] != null)
    {
        var currentId = obj[idKey]!.ToString();
        if (idMap.TryGetValue(currentId, out var newId))
        {
            obj[idKey] = newId;
        }
    }
    foreach (var kvp in obj.ToList())
    {
        ReplaceNodeIdsInJson(kvp.Value, idMap, idKey);
    }
}
else if (node is JsonArray arr)
{
    foreach (var item in arr)
    {
        ReplaceNodeIdsInJson(item, idMap, idKey);
    }
}
```

Метод виконує рекурсивний обхід JSON-структури документа та здійснює заміну ідентифікаторів вкладених об'єктів. Спочатку перевіряється, чи існує вузол для обробки. Якщо поточний елемент є JSON-об'єктом, система перевіряє наявність поля з ідентифікатором та, за потреби, замінює його на новий відповідно до словника відповідностей. Після цього рекурсивно обробляються

всі вкладені властивості об'єкта. У випадку, якщо поточний вузол є масивом, виконується послідовна обробка кожного його елемента.

Зображення, які більше не використовуються у документі, не видаляються одразу. Вони позначаються, як неактуальні та надалі очищаються за допомогою фоновому процесу. Для реалізації такого механізму використовується пакет Quartz, який дозволяє планувати виконання завдань на визначений момент часу [24]. Після запуску сервера планувальник відстежує час виконання завдання та автоматично запускає його відповідно до заданих параметрів. Такий підхід забезпечує видалення неактуальних записів як із бази даних, так і з хмарного сховища. Саме тому після створення копій пов'язаних сутностей необхідно модифікувати JSON-дерево документа, замінюючи старі ідентифікатори новими.

Приклад реєстрації запланованого завдання:

```
q.AddJobAndTrigger<CleanOrphanedImagesJob>(builder.Configuration);
```

У даній реалізації використано власне розширення для спрощення та уніфікації процесу реєстрації фонових завдань у Quartz.NET. Наведений метод розширення реалізує автоматичну конфігурацію завдання та тригера на основі узгоджених правил реєстрації часу виконання завдань:

Лістинг коду розширення для реєстрації залежності:

```
public static void AddJobAndTrigger<T>(this IServiceCollectionQuartzConfigurator quartz,
    IConfiguration config)
    where T : IJob, IQuartzConfig
    {
        string jobName = T.JobName;
        if (string.IsNullOrEmpty(jobName))
            throw new ArgumentNullException(jobName);

        var configKey = $"Quartz:{jobName}";
        var cronSchedule = config[configKey];

        if (string.IsNullOrEmpty(cronSchedule))
            throw new ArgumentNullException($"No Quartz.NET Cron schedule found for job in configuration at {configKey}");

        quartz.AddJob<T>(opts => opts.WithIdentity(jobName));

        quartz.AddTrigger(opts => opts
            .ForJob(jobName)
            .WithIdentity(jobName + "-trigger")
            .WithCronSchedule(cronSchedule));
    }
```

Даний підхід дозволяє значно зменшити кількість повторюваного коду та забезпечує централізовану конфігурацію фонових завдань. Час виконання

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		49

кожного завдання задається у файлі конфігурації appsettings.json у вигляді cron-виразу. Для кожного завдання обов'язковою є реалізація інтерфейсу IQuartzConfig, який визначає його унікальне ім'я через властивість JobName, що використовується для ідентифікації та зв'язування конфігурації із конкретним завданням.

Однією з функціональних можливостей вебплатформи є глобальний пошук, який забезпечує швидке знаходження користувачів, косметичних продуктів, колекцій та публікацій. Для забезпечення високої продуктивності та гнучкості пошукових запитів у системі було використано механізми оптимізації PostgreSQL, зокрема індекси типу GIN (Generalized Inverted Index) [25] та розширення pg_trgm.

Індекси GIN дозволяють ефективно виконувати пошук по текстових полях за принципом, які рядки містять певне значення у колонці. У свою чергу, розширення pg_trgm [26], він розбиває рядки на послідовності з трьох символів (триграми), що дозволяє виконувати нечіткий пошук та визначати ступінь схожості між рядками.

Також для підвищення ефективності роботи з базою даних у серверній частині застосунку використовується IDbContextFactory. Це рішення зумовлено тим, що DbContext в Entity Framework Core не є потокобезпечним, тому створення окремого контексту для кожного запиту дозволяє уникнути проблем конкурентного доступу та покращує стабільність системи.

У наведеному прикладі спочатку створюється окремий екземпляр DbContext за допомогою IDbContextFactory. Далі формується запит до бази даних із застосуванням фільтрації за видимістю сутностей, наявністю пов'язаних продуктів та збігами пошукового запиту.

Лістинг коду реалізація запиту до бази даних:

```
await using var context = await _contextFactory.CreateDbContextAsync(ct);

var query = context.Collections
    .AsNoTracking()
    .AsSplitQuery()
    .Include(c => c.TagCollections).ThenInclude(tc => tc.Tag)
    .Include(c => c.User)
    .Where(c => !c.IsPrivate && !c.User.IsPrivate)
    .Where(c => c.CollectionProducts.Any())
```

```

        .Where(c => EF.Functions.ILike(c.Title, $"%{searchTerm}%") ||
            EF.Functions.TrigramsAreSimilar(c.Title, searchTerm) ||
            c.TagCollections.Any(tc =>
                EF.Functions.ILike(tc.Tag.Name, $"%{searchTerm}%") ||
                EF.Functions.TrigramsAreSimilar(tc.Tag.Name, searchTerm)));

    var totalCount = await query.CountAsync(ct);

    var items = await query
        .OrderByDescending(c => EF.Functions.TrigramsSimilarity(c.Title,
            searchTerm))
        .ApplyPagination(pageNumber, pageSize)
        .ToListAsync(ct);

    return (items, totalCount);

```

Для реалізації пошуку використовується комбінація нечутливого до регістру порівняння ILIKE та триграмного аналізу, що дозволяє знаходити часткові та приблизні збіги. Сортування результатів виконується за ступенем схожості рядків, після чого застосовується пагінація для обмеження обсягу вибірки.

Висновки до другого розділу

У результаті виконання другого розділу було визначено основні вимоги користувачів до вебплатформи. На основі проведеного аналізу сформовано функціональні та нефункціональні вимоги, а також побудовано діаграму варіантів використання системи, у якій виділено три основні актори, що будуть взаємодіяти з системою. Це дозволило чітко визначити межі відповідальності кожного типу користувача та описати їхні сценарії взаємодії з програмним продуктом.

Отримані вимоги стали основою для подальшого проєктування архітектури програмного забезпечення та розробки діаграми класів, на якій було продемонстровано детальнішу організацію кожного з шарів серверної частини системи та структуру компонентів програми, їхні методи та взаємозв'язки між ними. Також було продемонстровано використання шаблонів проєктування, у межах яких обґрунтовано їхній вибір та описано їхню роль у системі.

Додатково було спроектовано структуру бази даних, яка включає тридцять шість таблиць та їхні взаємозв'язки. У межах моделі даних реалізовано різні типи зв'язків, зокрема зв'язок «один-до-одного», який використовується, наприклад, для створення чернеток у системі, «один-до-багатьох» для зв'язування

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		51

користувача з його косметичними колекціями; а також «багато-до-багатьох» між постами та тегами.

Визначення такої кількості сутностей було необхідним для забезпечення основної логіки системи, зокрема роботи з користувачами, косметичними колекціями, постами, коментарями та іншими пов'язаними об'єктами. Також описано призначення ключових таблиць та їхню роль у реалізації системи.

Для деталізації поведінки системи були побудовані діаграми активності та послідовності, вони дозволили формалізувати алгоритми роботи основних функціональних процесів. Демонстрація діаграм дала змогу відобразити послідовність взаємодії між компонентами системи та логіку виконання ключових операцій у межах користувацьких сценаріїв.

Окремо наведено приклади реалізації програмних рішень, які демонструють ключові механізми системи, зокрема налаштування фонових процесів із використанням запланованих задач. Це дозволило показати практичну реалізацію архітектурних рішень, описаних на попередніх етапах проєктування, та підтвердити їхню ефективність у межах розробленого програмного продукту.

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

**РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З
ВЕБПЛАТФОРМОЮ ДЛЯ СТВОРЕННЯ КОСМЕТИЧНИХ КОЛЕКЦІЙ
ТА ВЕДЕННЯ БЛОГУ**

3.1 Порядок встановлення та налаштування параметрів

Для того, щоб платформа була доступна іншим користувачам, було використано Azure App Service для розгортання серверної частини, а для клієнтський використано Vercel, а баз даних обрано Azure PostgreSQL Database.

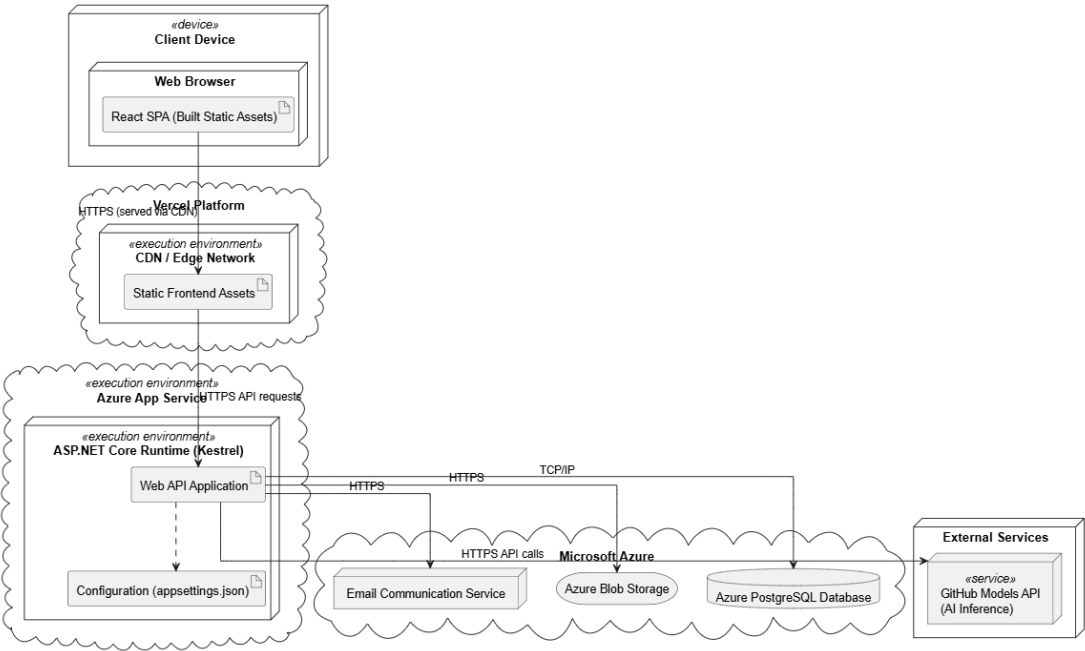


Рис. 3.1. Діаграма розгортання вебплатформи

Також як було згадано у минулих розділах, було використано Azure Blob Storage для збереження неструктурованих даних у системі, в даному випадку тільки зображення. А також для надсилання листів було використано Azure Email Communication Service.

Налаштування клієнтської частини вебплатформи було виконано за допомогою платформи Vercel, яка використовується для розгортання сучасних вебзастосунків. Розгортання проєкту здійснювалося через Vercel CLI, що дозволило автоматизувати процес білду та деплою без необхідності ручного налаштування кожного етапу.

Як було сказано вище, для серверної частини вебплатформи, було обрано Azure App Service [27]. Важливою перевагою Azure App Service є підтримка

автоматичного масштабування та інтеграція з іншими сервісами екосистеми Azure. Крім того, сервіс забезпечує централізоване керування конфігурацією застосунку та змінними середовища. Це дозволяє відокремити конфіденційні дані, такі як ключі доступу та рядки підключення до бази даних, від вихідного коду проєкту. Також App Service підтримує механізми моніторингу та діагностики, що спрощує пошук та усунення помилок під час експлуатації системи. Основні етапи для налаштування самого сервісу включав обирання середовища виконання, версії .NET та де він буде розташовуватися, найкраще обирати те місце, де знаходиться найближче сервер для швидкодії. Далі потрібно налаштувати змінні оточення, які будуть використовуватися під час роботи системи. Також, важливим етапом без якого сервер не буде приймати запити від клієнта є налаштування CORS, для того щоб клієнт і сервер могли спілкуватися між собою без проблем, там потрібно вказати дозволені адреси, які можуть звертатися до даного ресурсу системи.

База даних вебплатформи розгорнута на базі Azure Database for PostgreSQL Flexible Server, що є керованим хмарним сервісом для роботи з реляційними базами даних [28]. Також під час налаштування було підключено необхідні розширення PostgreSQL, які використовувалися під час локальної розробки та забезпечують коректну роботу функціональних можливостей системи. Підключення до бази даних здійснюється через захищене з'єднання, а автентифікація реалізована за допомогою рядка підключення, який містить необхідні параметри доступу до сервісу.

Налаштування надсилання електронних листів за допомогою Azure Email Communication Service складається з декількох етапів. Насамперед необхідно створити відповідний ресурс електронної пошти в середовищі Azure. Під час його створення налаштовується домен або піддомен, для цього було обрано Azure-managed domain, адже він є безкоштовним і він автоматично налаштовує всі потрібні сертифікати, який буде використовуватися як адреса відправника повідомлень.

Наступним кроком є прив'язка Email Communication Service до Azure Communication Services. Така інтеграція надає можливість виконувати надсилення електронних листів через програмний інтерфейс (REST API) [29], це дозволяє автоматизувати процес відправлення повідомлень без використання сторонніх поштових сервісів.

Під час налаштування Azure Blob Storage для зберігання неструктурованих даних було встановлено рівень публічного доступу для читання. Таке рішення дозволяє клієнтському застосунку без додаткових запитів до серверної частини отримувати доступ до зображень та відображати їх на вебплатформі. Також під час розробки використовувалися SAS-токени (Shared Access Signature) – це спеціальні захищені рядки, що додаються до URL-адреси ресурсу та надають обмежений у часі доступ до нього [30]. Також для організації файлів у сховищі використовуються контейнери, які дозволяють логічно групувати різні типи ресурсів. У межах проєкту окремий контейнер використовується для зберігання зображень в залежності від їх сфери застосування.

Використання SAS-токенів дозволило реалізувати пряме завантаження зображень із застосунку до Azure Blob Storage без передачі файлів через серверну частину. Необхідність такого рішення виникла через особливість роботи редактора постів. Оскільки редактор автоматично зберігає зміни через певний проміжок часу після припинення введення тексту користувачем, передавання всіх зображень через сервер при кожному оновленні контенту могло б суттєво збільшити навантаження на систему. Тому під час оновлення контенту сервер генерує SAS-токен із обмеженим терміном дії та визначеним доступом до одного blob. Додатково налаштовано вимогу використання виключно захищеного HTTPS-з'єднання.

Для інтеграції технологій штучного інтелекту в систему було використано GitHub Models, що надає доступ до різних великих мовних моделей через єдиний інтерфейс. Для підключення моделі до застосунку необхідно було створити Personal Access Token (PAT), який надає авторизований доступ до сервісу, розміщується він у змінних середовища, налаштованих у Azure App Service.

Після цього під час конфігурації Semantic Kernel отриманий токен використовується як ключ доступу до моделі.

3.2 Структура інтерфейсу вебплатформи для створення косметичної колекції та ведення блогу

Відкриваючи вебплатформу, користувач потрапляє на головну сторінку (рис. 3.2), де може ознайомитися з основними можливостями системи. На ній відображаються популярні публікації та косметичні колекції користувачів, які мали активність зі своїми об’єктами в період семи днів, це створено для того, щоб нові користувачі з’являлися у стрічці, нещодавно додані дописи, а також інструмент аналізу складу косметичних продуктів із використанням технологій штучного інтелекту.

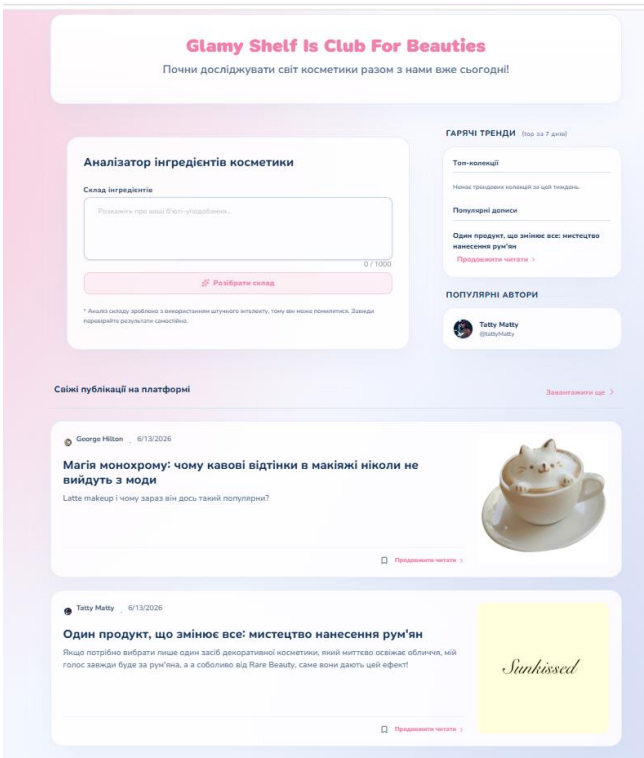


Рис. 3.2. Головна сторінка вебсайту

Для аналізу складу варто лише ввести перелік продуктів через кому, для того щоб ШІ міг його проаналізувати, в результаті аналізу користувач отримує вікно з результатами аналізу, де описано його безпечність та опис самого інгредієнту, якщо штучний інтелект, не знає цього інгредієнту, він просто залишає його як невідомий (рис. 3.3).

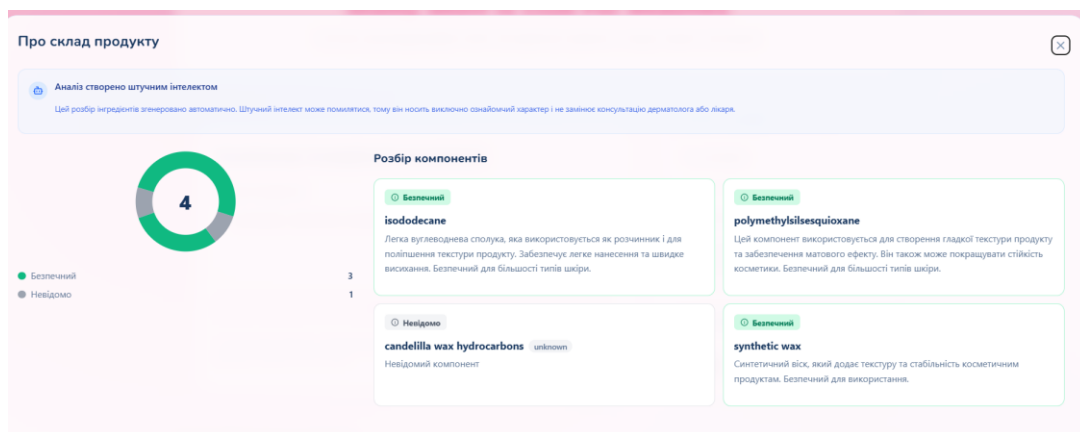


Рис. 3.3. Аналіз складу продукту

Для повноцінного використання функціональних можливостей системи, зокрема створення косметичних колекцій, публікацій та взаємодії з іншими користувачами, необхідно авторизуватися або зареєструвати новий обліковий запис. Після входу користувач отримує можливість налаштувати власний профіль відповідно до власних вподобань (рис. 3.4).

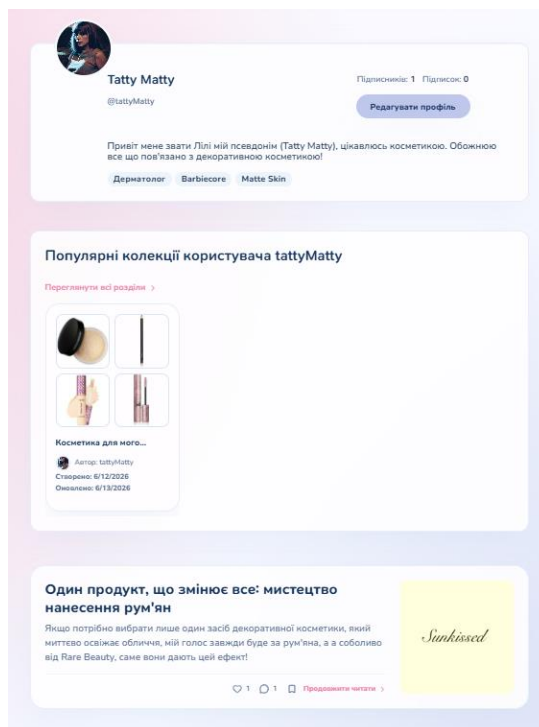


Рис. 3.4. Сторінка профілю користувача

У разі необхідності користувач може змінити рівень приватності власного профілю. Якщо профіль позначено як приватний, інші користувачі, які не є підписниками, не матимуть доступу до нових публікацій та колекцій власника профілю, а також не зможуть зберігати їх до розділу «Вибране» (рис. 3.5). Крім

того, новий контент такого користувача не відображатиметься на головній сторінці платформи.

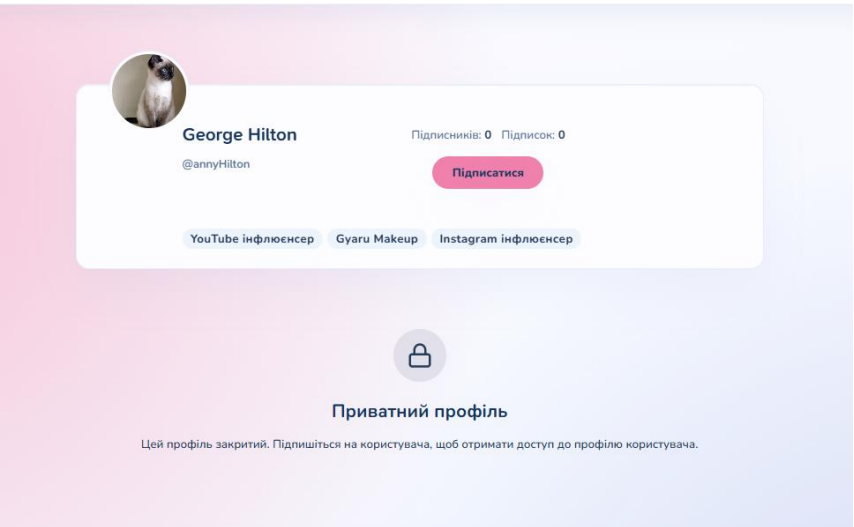


Рис. 3.5. Приватний профіль користувача

Для створення косметичної колекції користувачу необхідно перейти до розділу «Особистий простір», обравши відповідний пункт меню. Після цього потрібно створити новий розділ або обрати вже існуючий, у межах якого створюється нова косметична колекція. Після успішного створення колекції система автоматично перенаправляє користувача на її сторінку, де він може додавати косметичні продукти, змінювати налаштування видимості та редагувати інформацію про колекцію та видаляти (рис. 3.6).

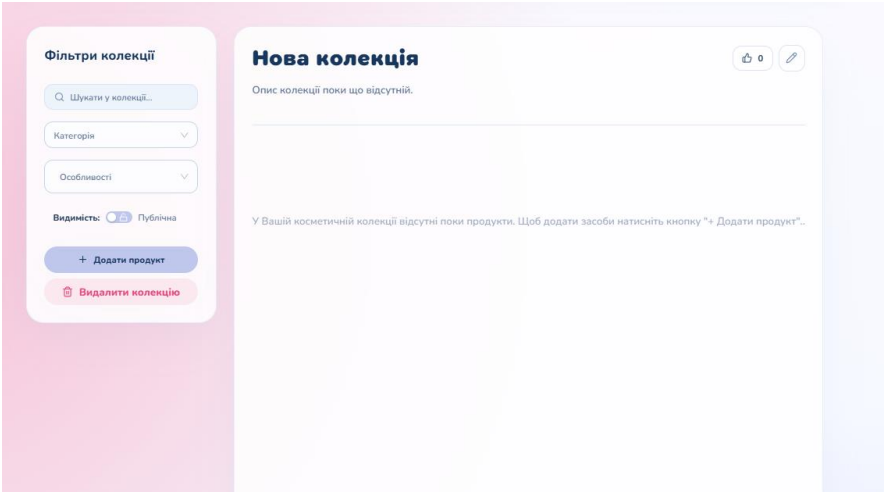


Рис. 3.6. Створена нова косметична колекція

Для додавання продукту до косметичної колекції користувач натискає кнопку «Додати продукт», після чого система перенаправляє його до каталогу

продуктів. На цій сторінці користувач може знайти необхідний продукт зі списку доступних, скориставшись пошуком або фільтрами (рис. 3.7).

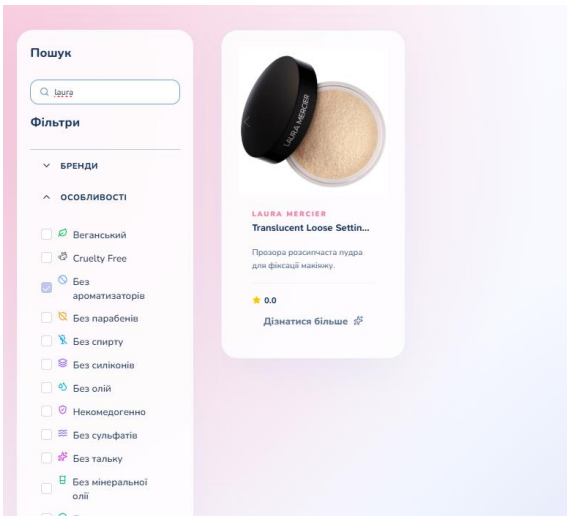


Рис. 3.7. Каталог продукції з використанням фільтрів

Далі користувач переходить на сторінку продукту, де може переглянути детальну інформацію про нього, залишити коментар, поставити оцінку, ознайомитися з аналізом складу та автоматично сформованим підсумком на його основі, а також переглянути категорії, до яких належить продукт (рис. 3.8).

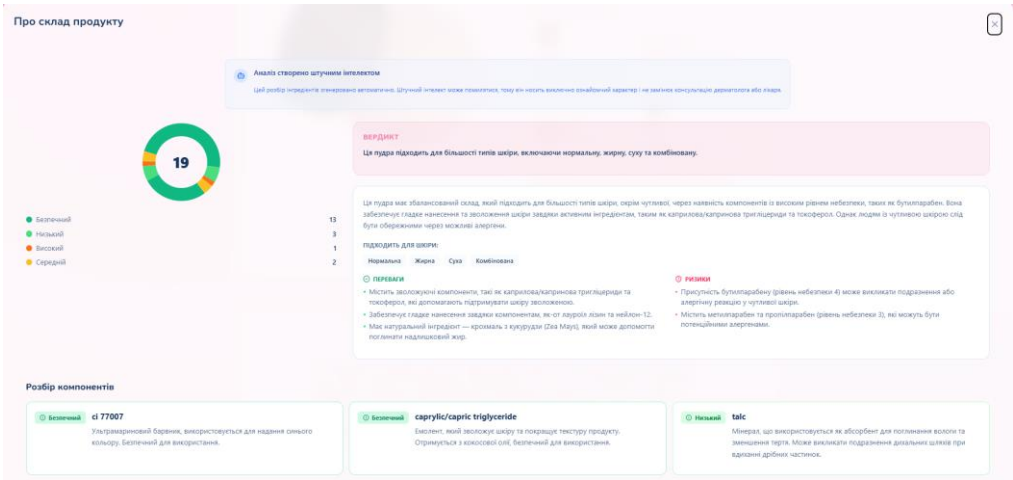


Рис. 3.8. Підсумок аналізу продукту

Також користувач має можливість залишати коментарі та відповідати на коментарі інших учасників платформи, а також прикріплювати фотографії до них. Функціональність коментування доступна для таких типів об'єктів, як косметичні колекції, пости та продукти (рис. 3.9).

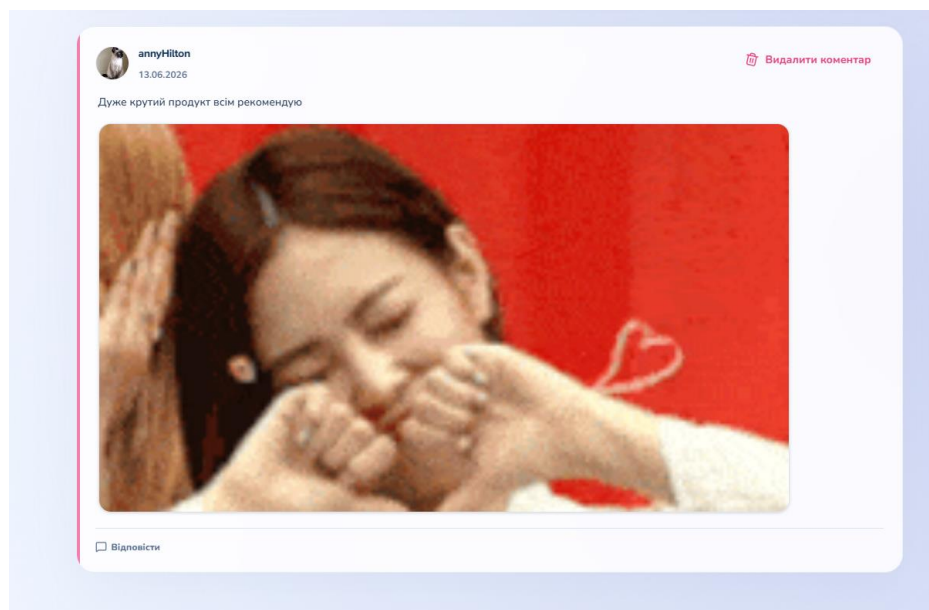


Рис. 3.9. Вигляд коментарю

Для додавання продукту до косметичної колекції користувачу необхідно обрати відповідний відтінок та натиснути кнопку «Додати до полицки». Після цього у пошуку відображається список доступних косметичних колекцій, з якого користувач обирає необхідну для додавання продукту (рис. 3.10).

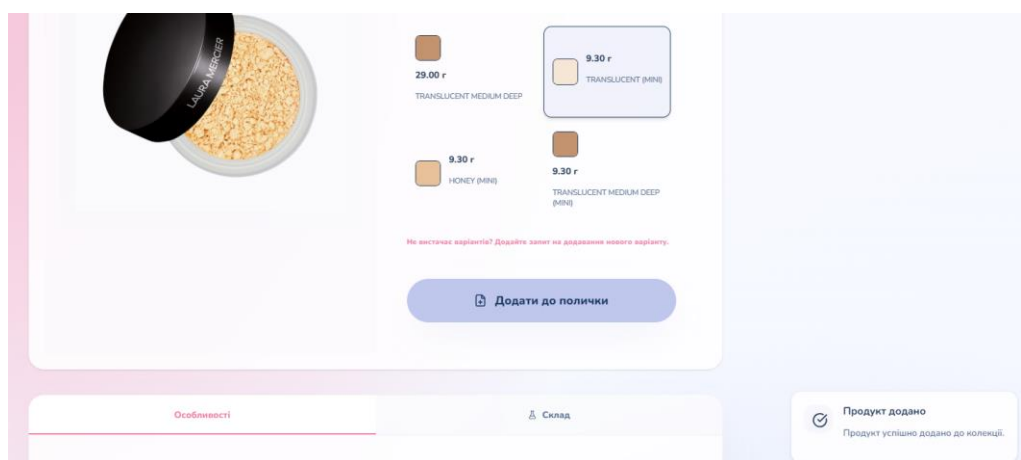


Рис. 3.10. Результат виконання додавання продукту до колекції

Інший користувач може переглядати косметичні колекції інших учасників платформи за умови, що вони є публічними, а профіль автора є відкритим. У випадку приватного профілю доступ до колекцій надається лише користувачам, які підписані на власника профілю. У межах перегляду колекції користувач може залишати коментарі, ставити реакції, а також додавати колекцію до секції «Вибране» шляхом вибору відповідного розділу (рис. 3.11).

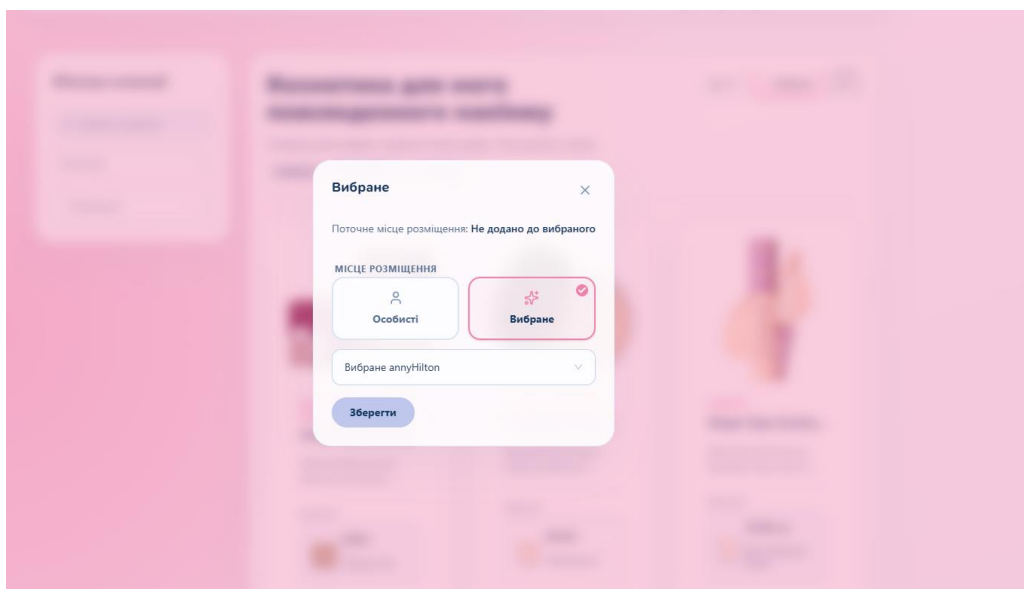


Рис. 3.11. Додавання колекції у секцію «Вибране»

За замовчуванням під час створення профілю для користувача створюється розділ у «Вибране» за замовчуванням. Цей розділ видалити чи змінити не можливо, в ньому також можна створювати підрозділи для його організації (рис. 3.12).

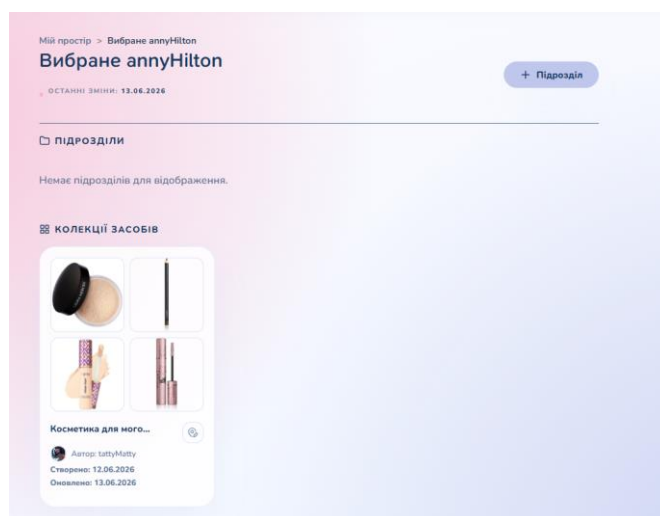


Рис. 3.12. Вигляд розділу за замовчуванням

Також у разі додавання іншими користувачами косметичної колекції до розділу «Вибране», її автор отримує відповідне сповіщення. Користувач має можливість переглянути деталі сповіщення та перейти до колекції, яка була додана до збережена іншим користувачем (рис. 3.13).

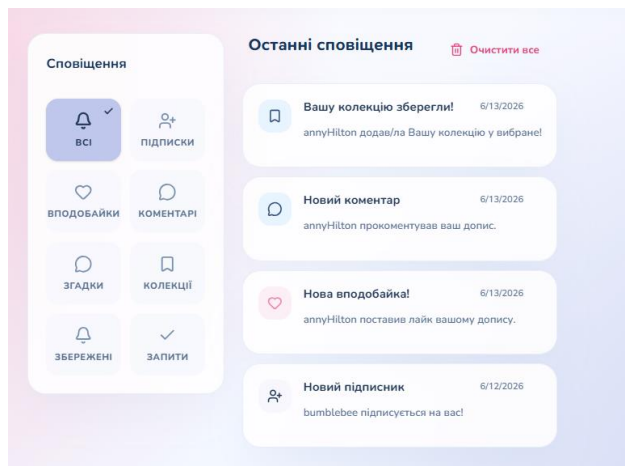


Рис. 3.13. Список сповіщень

У списку сповіщень (рис. 3.13) усі елементи є інтерактивними. Наприклад, у випадку отримання сповіщення про коментар або відповідь на нього, користувач може перейти до відповідної гілки обговорення та переглянути повний контекст коментаря, до якого було залишено відповідь (рис. 3.14).

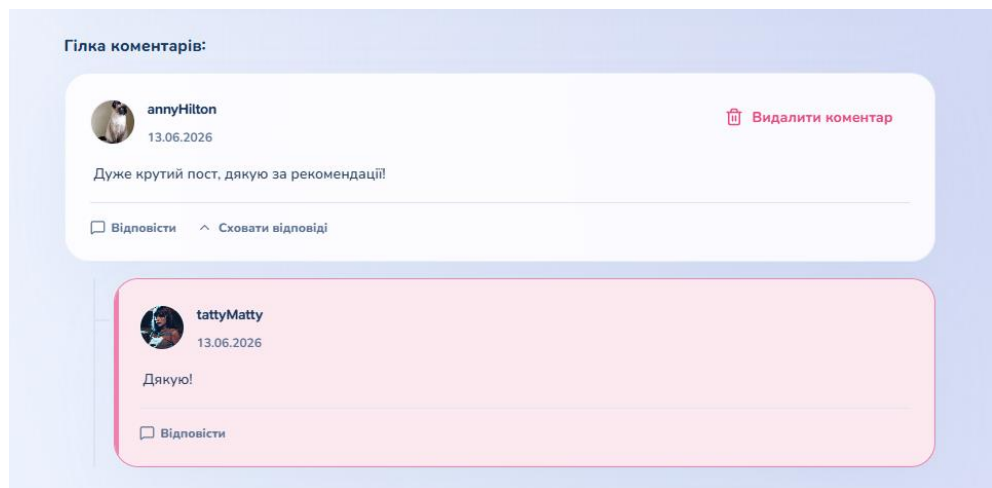


Рис. 3.14. Перегляд гілки коментарів

Майже всі сповіщення, окрім запитів на підписку мають свій час життя (30 днів), далі коментарі видаляються під час виконання фонових процесів, що був налаштований через Quartz. Також у випадку, коли колекція містить велику кількість продуктів реалізовано Keyset pagination (пагінацію за ключем). Для забезпечення більш плавної взаємодії користувача з інтерфейсом, створюючи ефект послідовного перегляду контенту (рис. 3.15).

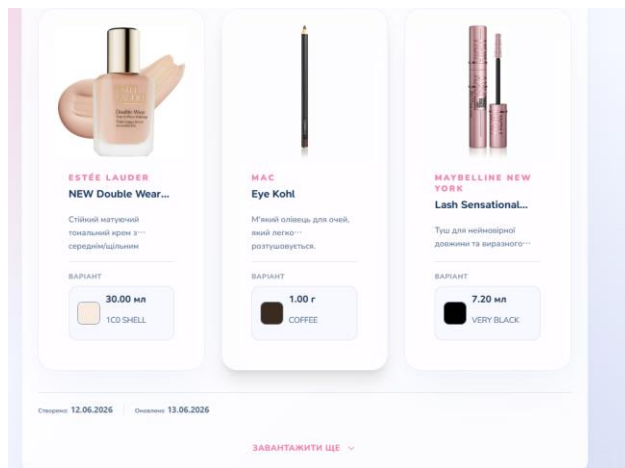


Рис. 3.15. Вигляд пагінації за ключем

Повертаючись до секцій, їх можна переміщувати по різних розділах користувача, в залежності від типу розділу, якщо це особистий розділ, його можна переміщувати лише в його межах, розділ за замовчування переміщувати не можливо (рис. 3.16).

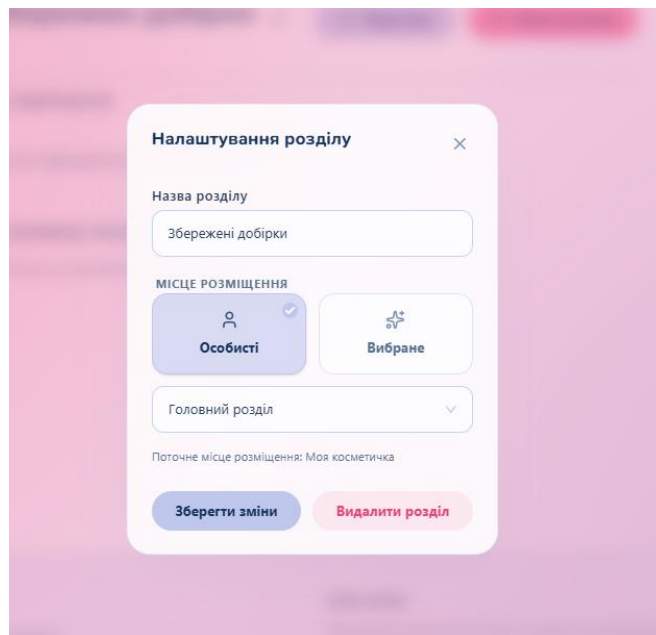


Рис. 3.16. Переміщення розділу в інший

За таким самим принципом можна переміщати колекції по різних розділам у особистому просторі користувача (рис. 3.17). Різниця лише є та що користувач не може перемістити колекцію у кореневий розділ.

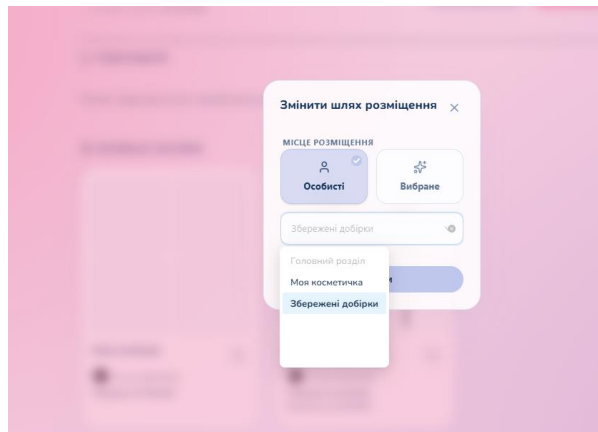


Рис. 3.17. Переміщення колекції у новий розділ

У випадку видалення розділу всі дочірні розділи автоматично переміщуються на рівень вище в ієрархії. Косметичні колекції, що містяться у видаленому розділі, повинні бути перенесені до іншого відповідного розділу користувача. Таке рішення реалізовано для запобігання розміщенню колекцій у кореневій (головній) секції (рис. 3.18).

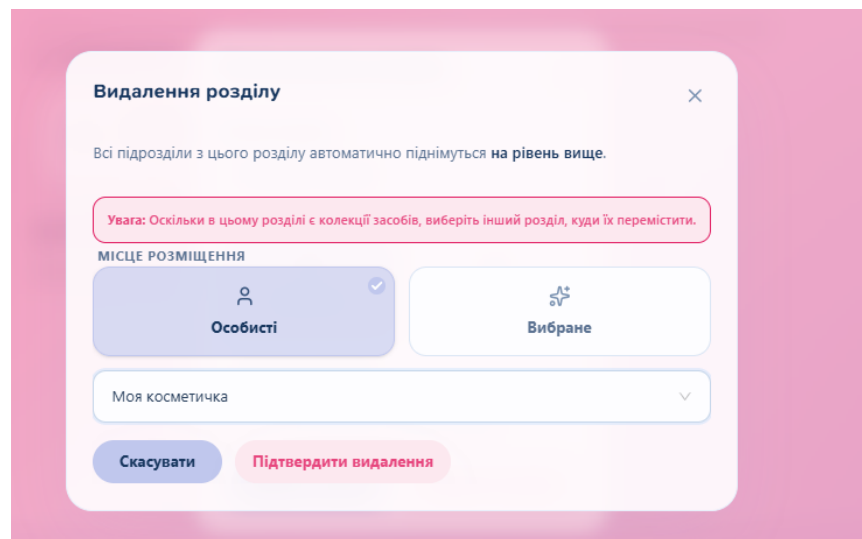


Рис. 3.18. Видалення розділу

Однією з основних функціональних можливостей вебплатформи є створення та управління публікаціями (постами), які слугують засобом обміну інформацією між користувачами та формування контенту системи (рис. 3.20).

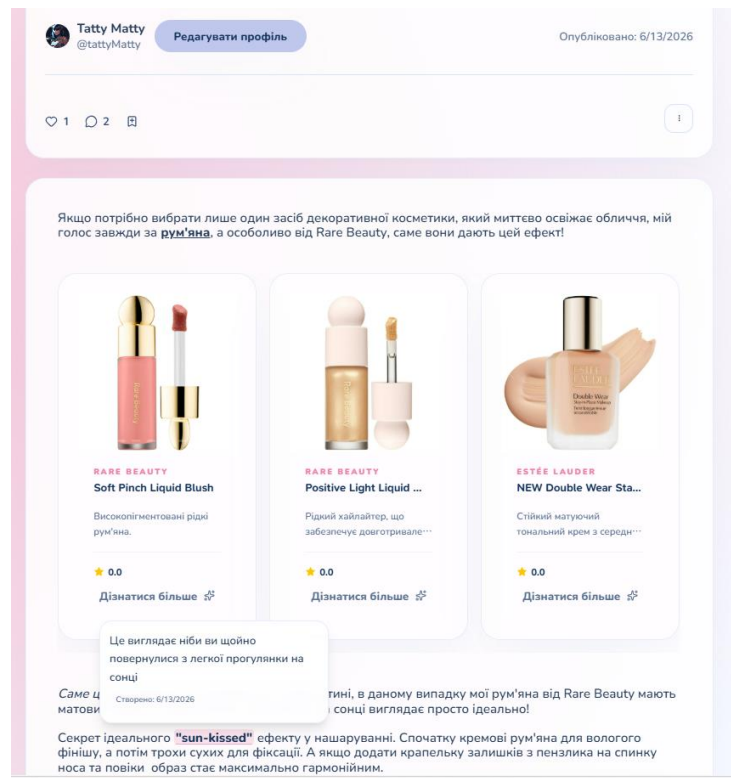


Рис. 3.20. Зовнішній вигляд опублікованого посту

Для створення власних постів користувач може скористатися кількома способами: через головне меню або шляхом переходу до розділу «Мої пости» та натискання кнопки «Створити новий пост» (рис. 3.21). На сторінці «Мої пости» користувач також має можливість переглядати свої публікації відповідно до їхнього статусу(чернетки, опубліковані та заархівовані пости).

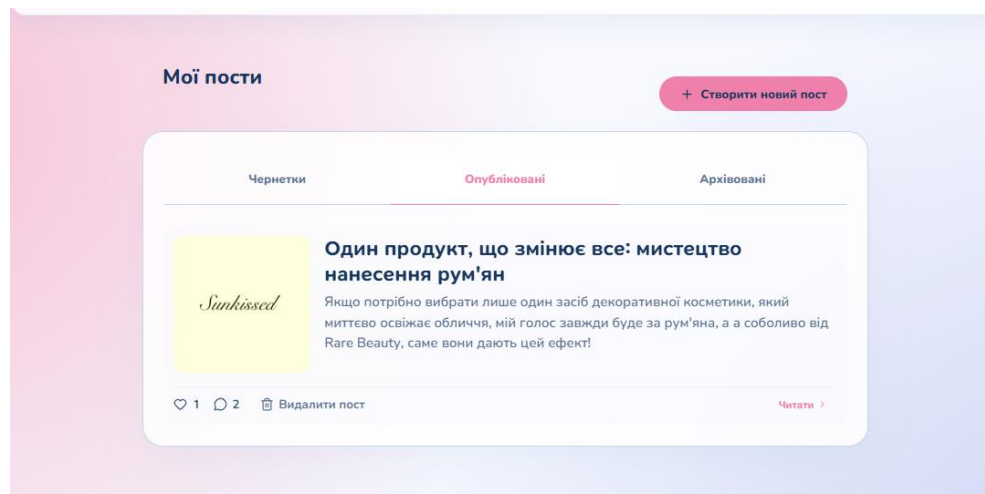


Рис. 3.21. Вигляд сторінки мої пости

Після створення нового посту користувач автоматично перенаправляється на сторінку редактора (рис. 3.22), де йому необхідно заповнити основні поля заголовок посту та основний контент.

Рис. 3.22. Вигляд створеної чернетки посту

Даний редактор підтримує (рис. 3.22) широкий набір функціональних можливостей, зокрема додавання косметичних колекцій і продуктів, вставку зображень, прасинг деяких видів посилань та перетворення їх у об'єкти, згадувати інших користувачів, а також створення нотаток до окремих слів або фрагментів тексту. Нотатки можуть бути як публічними, так і приватними (приватні нотатки відображаються лише в режимі чернетки або заархівованого посту).

Також реалізовано базові інструменти форматування тексту, такі як курсив, жирне виділення, підкреслення, створення підзаголовків, списків та вирівнювання тексту (рис. 3.23).

Рис. 3.23. Наповнена чернетка користувача

У разі необхідності продовження роботи над постом користувач може тимчасово залишити сторінку редактора. Після повернення до системи він має можливість відкрити незавершений пост у розділі «Мої пости» та продовжити його редагування, обравши відповідну чернетку зі списку (рис. 3.24).

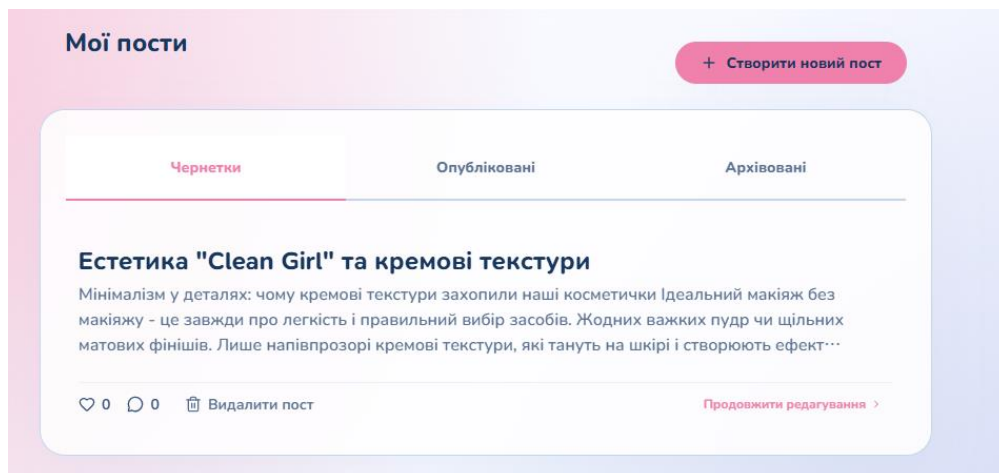


Рис. 3.24. Вигляд чернетки

Після завершення редагування користувач заповнює форму для попереднього перегляду посту (рис. 3.25). У цій формі передбачена можливість додавання зображення, короткого опису та тегів, які використовуються для подальшого пошуку публікації в системі. Короткий опис може бути заданий користувачем вручну або сформований автоматично на основі перших рядків тексту посту.

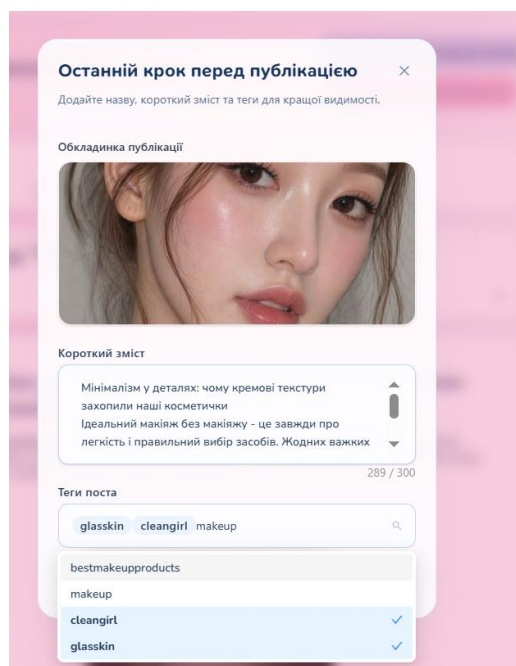


Рис. 3.25. Форма для заповнення попереднього перегляду

Користувачі, що були згадані у дописі, отримують відповідне сповіщення про те що їх там було відмітили (рис. 3.26). Згадана людина може перейти на публікацію і поглянути де саме її було згадано.

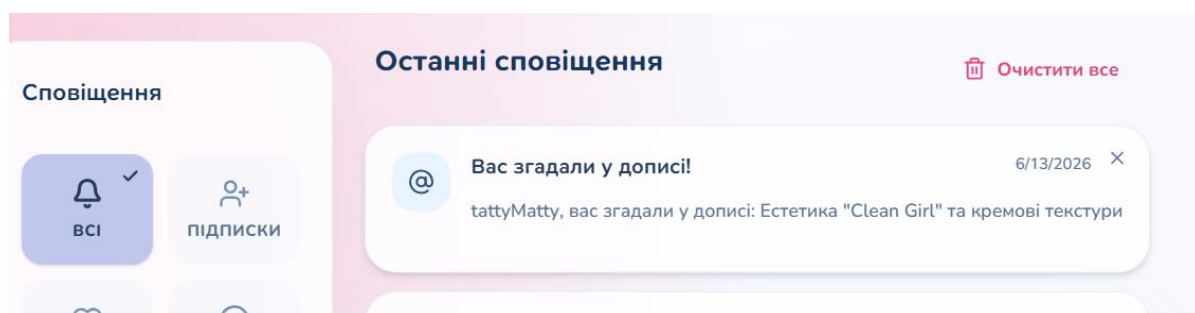


Рис. 3.26. Вигляд сповіщення про згадку

Якщо користувачу потрібно відредагувати вже опублікований пост, йому достатньо у дропдаун меню обрати «Редагувати» і створиться копія поточного посту який було обрано для редагування (рис. 3.27). Дану копію можна видалити за бажання користувача, якщо у цьому є необхідність.

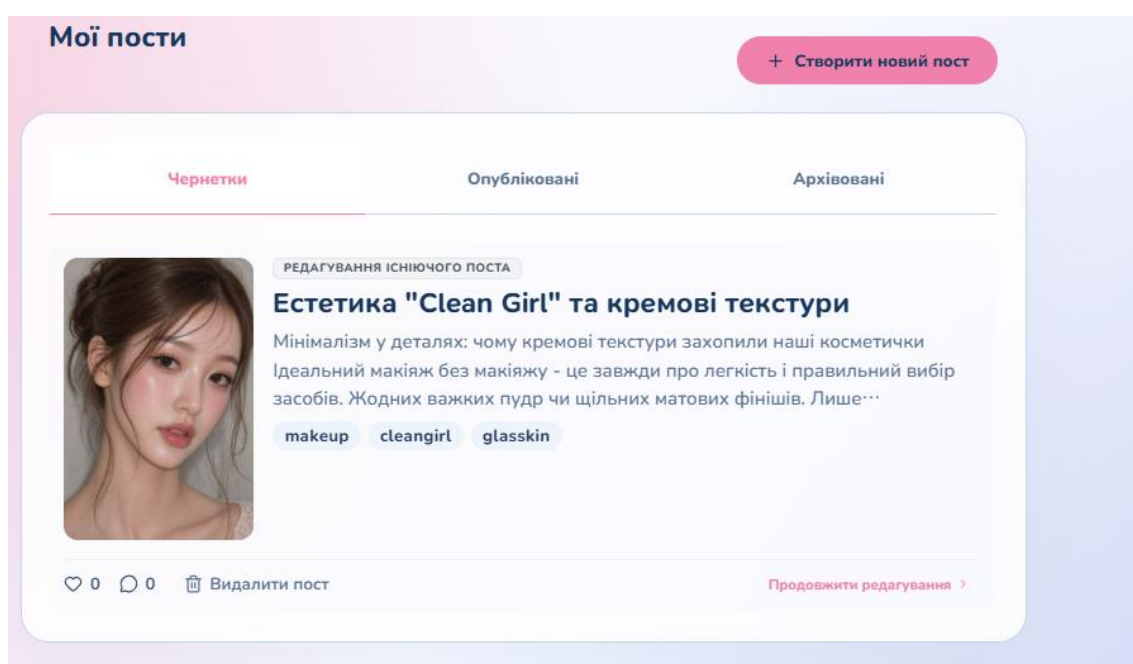


Рис. 3.27. Редагування існуючого посту

Після його редагування, відредагований допис користувача зливається з оригінальним дописом.

Також, інші користувачі, можуть зберігати допиши інших користувачів до збережених, де можна пізніше їх переглянути (рис. 3.28).

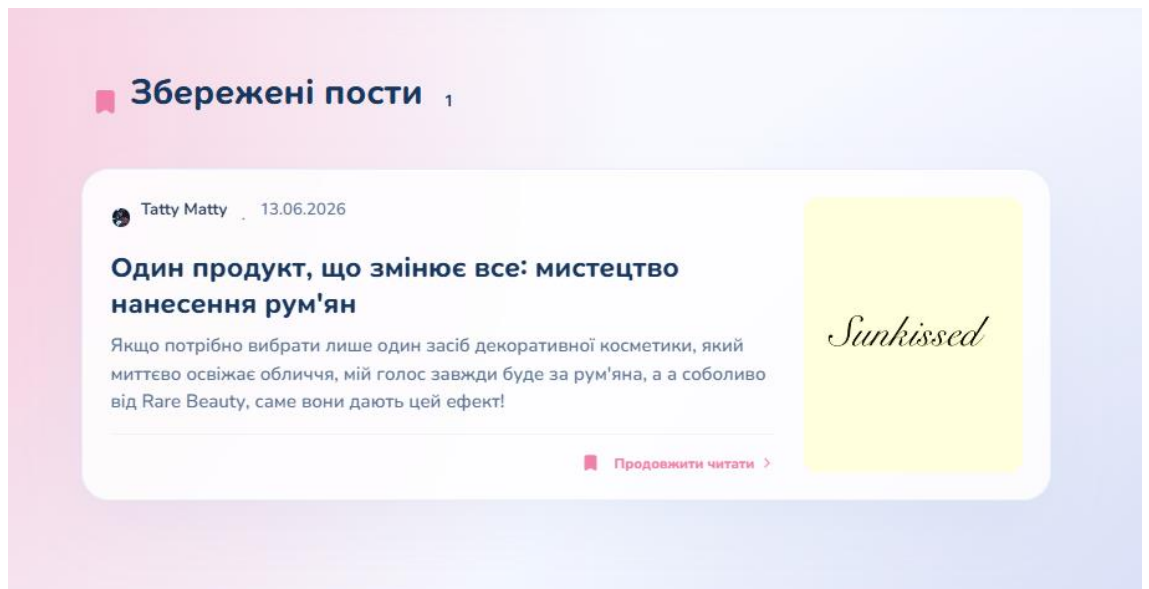


Рис. 3.28. Перегляд збережених постів користувача

У випадку приватного профілю користувача підписники мають можливість додавати його колекції до розділу «Вибране», тоді як інші користувачі не бачать інформацію про збереження таких колекцій у профілі учасника, що зберіг її (рис. 3.29).

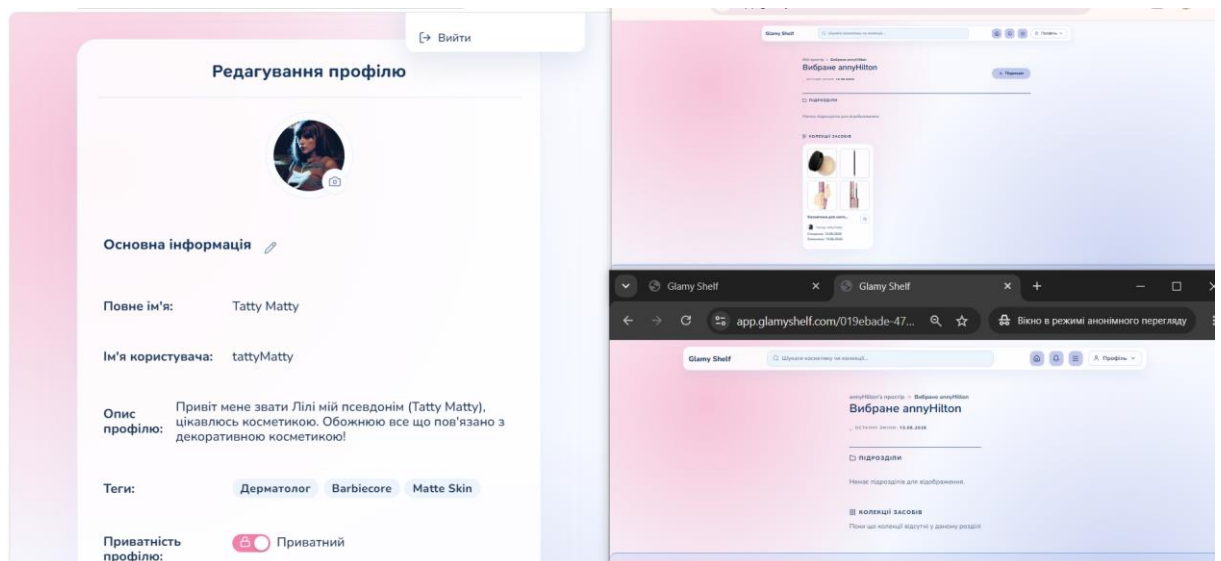
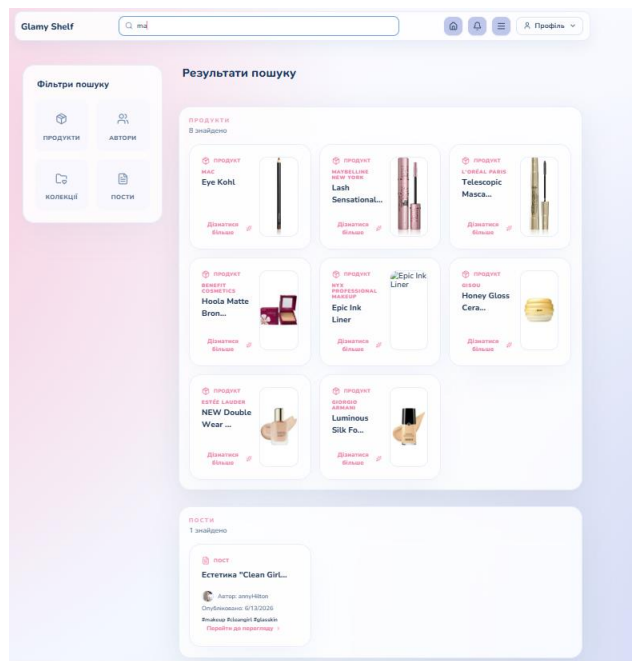


Рис. 3.29. Вигляд користувача що зберіг та того хто не є підписником

Можна також шукати різні об'єкти на платформі використовуючи глобальний пошук, що знаходиться у навігаційній панелі. Він шукає за різними характеристиками кожного об'єкту, наприклад, для дописів, використовується як пошук за тегами так і за заголовком. Отримані результати можна фільтрувати за їхнім видом, наприклад, за продуктом чи користувачем (рис. 3.30).



Якщо під час пошуку товарів в каталозі, не було знайдено відповідний продукт або варіант продукту, то можна відправити запит на додавання їх на платформу (рис. 3.31).

Форма для створення запиту (продукт) ×

ФОТОГРАФІЇ ЗОВНІШНЬОГО ВИГЛЯДУ ПРОДУКТУ



Назва продукту

Назва бренду

Опис варіанту

Хочу додати цей продукт на платформу бо не знайшла

Закласти

Надіслати запит

50 / 500

Рис. 3.31. Запит на додавання продукту

Після надсилання запиту він відображається на сторінці адміністратора «Додавання продуктів», де може бути опрацьований (рис. 3.32). Для зручності роботи передбачено можливість фільтрації запитів за типом, зокрема на додавання нового продукту або нового варіанта існуючого продукту.

					ІІЗ.КР.Б – 121 – 26 – ІЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

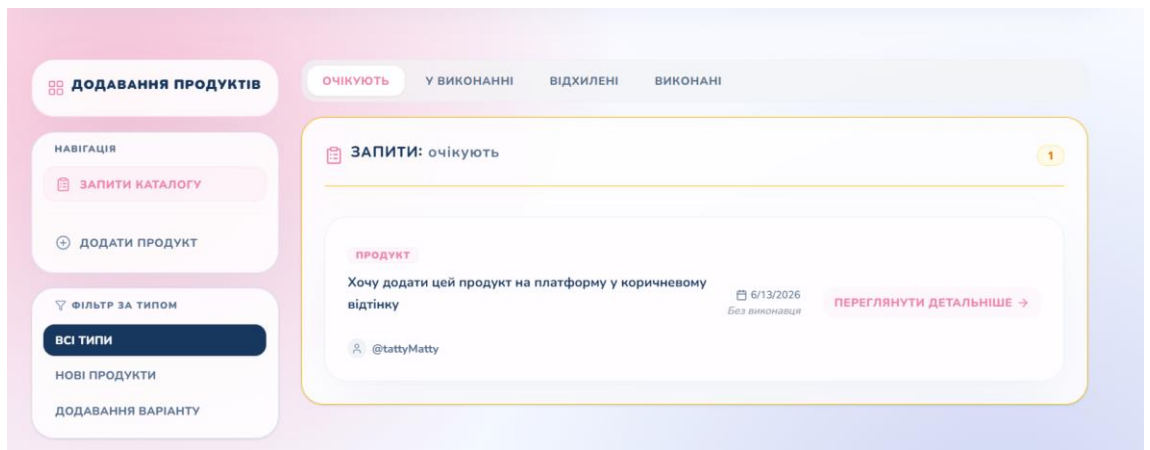


Рис. 3.32. Сторінка на додавання продуктів

Після створення користувачем запиту на додавання продукту він потрапляє до відповідного розділу адміністративної панелі (рис. 3.33). Для спрощення опрацювання запитів реалізовано механізм їх фільтрації за категоріями, зокрема на додавання нового продукту або нового варіанта продукту.

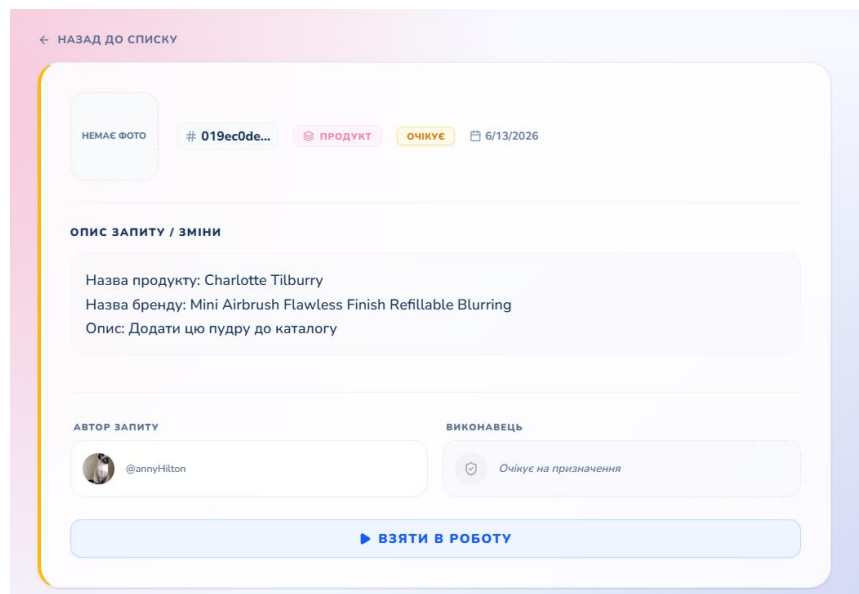


Рис. 3.33. Вигляд деталей запиту, запит на додавання продукту

Форма обробки запиту на додавання варіанта продукту має дещо інший вигляд (рис. 3.34). Для прискорення процесу опрацювання таких запитів адміністратору доступний швидкий перехід до сторінки з продуктом, до якого прив'язано відповідний запит.

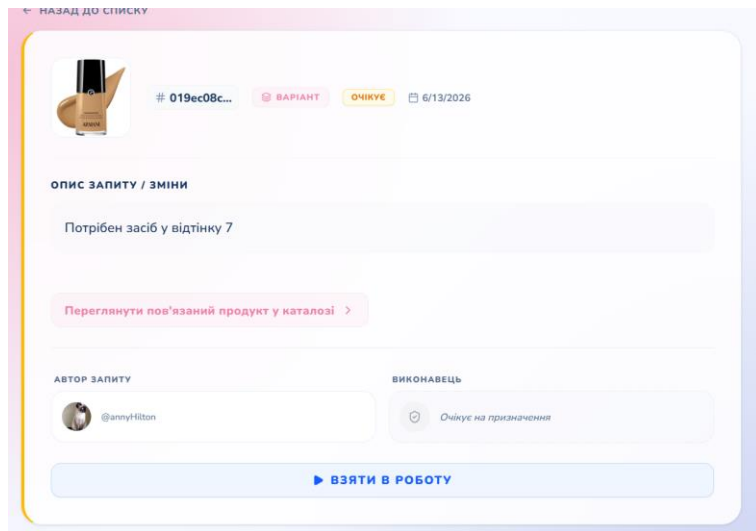


Рис. 3.34. Вигляд запиту на додавання варіанту

Якщо адміністратор приймає рішення відхилити запит, він повинен вказати причину відхилення.

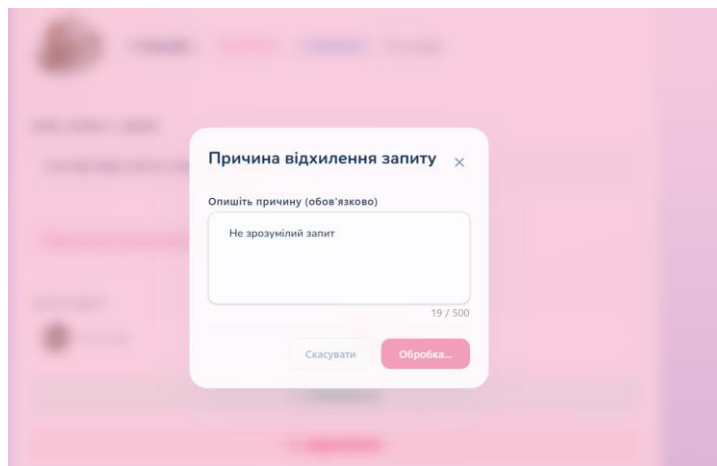


Рис. 3.35. Відхилення запиту на додавання

Після цього користувачу автоматично надсилається повідомлення на електронну пошту із зазначенням причини, а також створюється сповіщення на платформі про те, що його запит було відхилено (рис. 3.36).

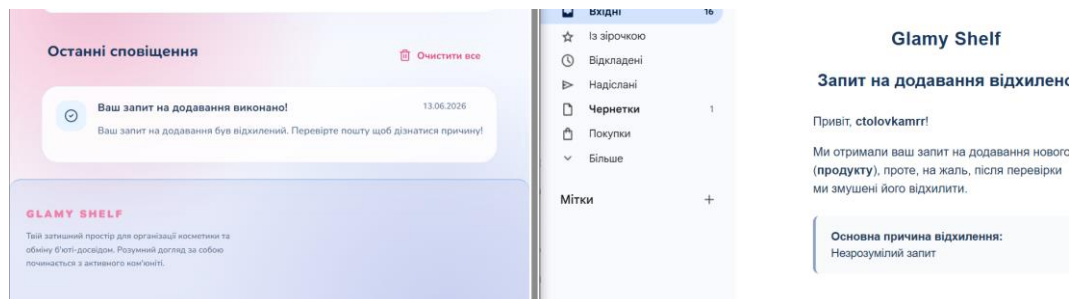


Рис. 3.36. Результат відхиленого запиту

Обравши в роботу запит на додавання продуктів, адміністратор може скористатися формою, в ній потрібно вказати назву продукту, обрати бренд чи є в продажі та інші характеристики продукту (рис. 3.37).

Рис. 3.37. Форма для додавання продуктів на платформу

Після додавання нового продукту адміністратор повинен обрати створений продукт через пошук та прив'язати його до відповідного запиту, підтвердивши його виконання. У випадку запиту на додавання варіанта продукту достатньо натиснути кнопку «Виконано» (рис. 3.38), після чого запит буде позначено як оброблений.

Рис. 3.38. Форма обробки запиту із прив'язаним продуктом що був доданий

Після успішного виконання запиту, користувач отримує сповіщення, за те що варіант чи продукт було додано на платформу, натиснувши на нього користувач буде направлений на відповідний продукт (рис. 3.39).

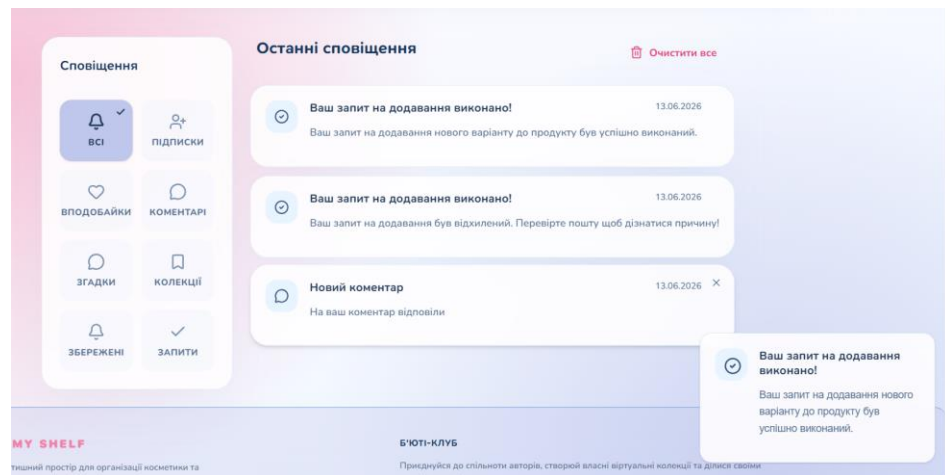


Рис. 3.39. Сповіщення про успішне виконання запиту

3.3 Тестування роботи програмного комплексу

У процесі розробки вебплатформи важливим етапом стало тестування програмного комплексу, яке дозволило перевірити коректність роботи основних функціональних модулів, виявити потенційні помилки та оцінити стабільність системи під час виконання типових сценаріїв використання. Окрему увагу було приділено перевірці взаємодії між клієнтською та серверною частинами, а також коректності обробки запитів до бази даних.

Тестування програмного забезпечення проводилося у форматі ручного тестування із використанням допоміжних інструментів. Зокрема, для перевірки коректності роботи API-ендпоінтів застосовувався інструмент Scalar, який забезпечує зручний інтерфейс для формування та виконання HTTP-запитів, а також перегляду відповідей сервера.

Додатково було проведено тестування адаптивності інтерфейсу користувача за допомогою вбудованих інструментів розробника у різних браузерах.

Окремим етапом перевірки стало тестування коректності зберігання конфігураційних та секретних даних у відповідних середовищах. Зокрема, у клієнтській частині використовувалися змінні середовища, а на серверній частині механізм User Secrets [31] для ASP .NET Core (середовище Dev), для розгонутого застосунку використовувалися змінні оточення APP Service.

Також було виконано перевірку валідності даних як на стороні клієнта, так і на стороні сервера, що дозволило забезпечити правильність збереження інформації системи. Окремо враховувалися механізми контролю доступу, зокрема перевірка прав користувача на читання даних відповідно до налаштувань приватності профілю.

Одним із важливих етапів тестування було забезпечення коректності зв'язків між розділами під час їх переміщення. Необхідно було гарантувати відсутність циклічних залежностей, які можуть виникати у випадку переміщення розділу в один із його дочірніх елементів. Для цього здійснювалася перевірка того, що ідентифікатор розділу, до якого виконується переміщення, не належить до нащадків поточного розділу. Також було реалізовано перевірки, що забезпечують переміщення особистих та вибраних колекцій виключно в межах відповідної секції, у якій вони були створені.

Для косметичних колекцій було проведено тестування, яке включало перевірку заборони додавання дублікатів до однієї колекції. Такий підхід дозволяє уникнути повторного створення однакових елементів у межах колекції та забезпечує цілісність даних. Окремо перевірялася умова створення косметичних колекцій лише в межах секцій відповідного типу. Зокрема, створення колекції дозволяється виключно у приватних секціях користувача, що забезпечує коректне розмежування між публічними та особистими структурами системи.

Оскільки у системі реалізовано механізм сповіщень, які не лише надсилаються користувачу, але й зберігаються у базі даних, було необхідно забезпечити коректність їх створення та обробки. Окрему увагу приділено запобіганню надмірній генерації сповіщень одним користувачем для іншого, що могло б призвести до спаму та надмірного навантаження на базу даних. Для цього було впроваджено механізм обмеження частоти створення сповіщень, який встановлює мінімальний інтервал часу між повторними сповіщеннями від одного користувача до іншого. Це дозволяє зменшити навантаження на систему та забезпечити більш стабільну роботу сервісу сповіщень.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		75

Оскільки у вебплатформі реалізовано власний редактор на базі Slate.js, окрему увагу під час тестування було приділено перевірці коректності побудови JSON-дерева документа. Це критично важливо для стабільної роботи механізму позиціонування каретки та роботи з текстовими вузлами у фреймворку [32]. Зокрема, перевірялася наявність обов'язкових порожніх текстових вузлів у межах блокових та void-елементів, відсутність яких призводить до некоректного позиціонування курсора та збоїв у роботі редактора.

Крім того, перевірялися переходи між різними статусами постів. Для реалізації цієї логіки використано підхід Машини Станів [20], який дозволяє контролювати допустимі переходи між станами сутності та виконання відповідних методів під час зміни статусу. У процесі тестування було перевірено коректність переходів, а також виконання методів, що відповідають за зміну стану посту. Машина Станів використовувалася не тільки для обробки станів дописів, а також для обробки статусів записів.

Важливим етапом тестування також стала перевірка коректності визначення авторства для різних об'єктів системи. Це необхідно для забезпечення правильного розмежування прав доступу та гарантування того, що користувачі взаємодіють лише з тими даними, до яких вони мають відповідні дозволи. Перевірялося, щоб кожен об'єкт у системі мав чітко визначеного власника, а операції редагування, видалення та перегляду виконувалися виключно відповідно до встановлених правил авторизації.

Через те, що у системі реалізовано механізм глобального пошуку по всій платформі, важливим етапом тестування стало забезпечення коректного обмеження видимості даних, що не повинні бути доступні для загального перегляду. Зокрема, було необхідно гарантувати, що у результати пошуку не потрапляють приватні колекції, а також дані користувачів із закритими профілями та пов'язані з ними об'єкти.

Також було перевірено коректність роботи фонових процесів, які використовуються для очищення бази даних від застарілих або непотрібних

даних. Окремо тестувалася ініціалізація та одноразове виконання запланованих задач, з метою запобігання їх повторного запуску або дублювання.

Оскільки у системі реалізовано два типи користувачів, було проведено тестування коректності розмежування прав доступу до функціональних можливостей платформи. Зокрема, перевірялося, щоб звичайний користувач не мав доступу до перегляду запитів, що надсилаються іншими користувачами до системи. Також було перевірено відсутність можливості виконання адміністративних дій, зокрема додавання продуктів та їх варіантів, а також ініціалізації оновлення інгредієнтів у системі для користувачів без відповідних прав доступу.

Висновки до третього розділу

У результаті виконання третього розділу було продемонстровано основні сервіси та налаштування, які використовувалися під час розгортання програмного продукту. Зокрема, описано процес конфігурації серверної частини, бази даних та клієнтського застосунку, а також як налаштовувалися додаткові компоненти системи, наприклад, для зберігання неструктурованих даних чи відправки електронних листів на користувацьку пошту, а також їхню взаємодію в межах загальної архітектури системи.

Також у межах даного розділу було представлено користувацький інтерфейс вебплатформи та було показано ключові сценарії використання системи, такі як створення та редагування контенту, робота з косметичними колекціями, а також взаємодія з іншими користувачами через коментарі, реакції та збереження контенту.

Обов'язковим етапом було проведення перевірки стабільності та працездатності програмного продукту в умовах різних сценаріїв використання. Проведені роботи підтвердили, що розроблювана веб платформа є готовою для використання іншими користувачами.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		77

ВИСНОВКИ

У результаті виконання випускної кваліфікаційної роботи бакалавра було спроектовано та реалізовано вебплатформу для створення косметичних колекцій та ведення блогу користувачів.

На перших етапах роботи було, проаналізовано проблеми з якими зіштовхуються користувачі, після чого було сформовано перелік завдань, які необхідно було реалізувати в межах проєкту, а також досліджено існуючі аналоги, що надають подібний функціонал та визначено їхні особливості, переваги та недоліки. На основі проведеного аналізу були визначені вимоги до програмного продукту, обрано архітектуру системи та підібрано відповідний технологічний стек для його реалізації.

Перед обранням остаточного стеку технологій було проведено порівняння з іншими технологіями, що є популярними та активно підтримуваними у середовищі. У висновку було обрано ASP.NET Core Web API для серверної частини, для побудови інтерфейсу користувача застосовано бібліотеку React, PostgreSQL для збереження даних, а також застосовано хмарні сервіси Microsoft Azure, які використовуються для зберігання, медіа-даних та надсилання електронних листів користувачам. Додатково було інтегровано механізми роботи зі штучним інтелектом за допомогою Semantic Kernel для великих мовних моделей.

На наступному етапі було визначено вимоги користувачів, на основі яких сформовано функціональні та нефункціональні вимоги, визначено основних акторів системи та побудовано діаграму варіантів використання програмного продукту. Обов'язковим етапом стало проєктування діаграми класів та структури бази даних, що відображають архітектуру системи та взаємодію даних між собою.

Подальший етап включав проєктування та реалізацію алгоритмів роботи системи, що супроводжувалося побудовою діаграм активностей та послідовності. Також було продемонстровано фрагменти програмного коду, які ілюструють роботу окремих компонентів системи.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		78

Окремо було описано процес розгортання застосунку та використані платформи. Зокрема, клієнтська частина була розгорнута на платформі Vercel, інтеграція зі штучним інтелектом здійснювалася через GitHub Models, а інші компоненти програмного продукту були розміщені у хмарному середовищі Microsoft Azure, а саме серверна частина була розміщена в Azure App Service, для зберігання медіафайлів системи було використано Azure Blob Storage, надсилання електронних листів відбувалося через Azure Email Communication та база даних була розгорнута на Azure Database for PostgreSQL Flexible Server.

Також було продемонстровано загальну структуру розробленої вебплатформи та роботу її основного функціоналу.

У висновку розроблена вебплатформа допомагає користувачам організовувати власні косметичні продукти, а також створює інтерактивне середовище для взаємодії та обміну досвідом між користувачами, забезпечуючи зручний інструмент для ведення персоналізованих косметичних колекцій.

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		79

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Beauty industry statistics 2026 [cosmetic market size]. DemandSage. URL: <https://www.demandsage.com/beauty-industry-statistics/> (дата звернення: 30.01.2026).
2. Pinterest. Pinterest. URL: <https://www.pinterest.com/> (дата звернення: 30.01.2026).
3. Beauty Insider Community By Sephora. Sephora Community. URL: <https://community.sephora.com/> (дата звернення: 30.01.2026).
4. Перевірка складу косметики онлайн - Український розшифровувач інгредієнтів косметики. SafetyMakeup. URL: <https://safetymakeup.com.ua> (дата звернення: 30.01.2026).
5. What Is a REST API (RESTful API)?. IBM. URL: <https://www.ibm.com/think/topics/rest-apis> (дата звернення: 13.06.2026).
6. Powell P., Smalley I. Monolithic vs. Microservices architecture. IBM. URL: <https://www.ibm.com/think/topics/monolithic-vs-microservices> (дата звернення: 20.02.2026).
7. Normand E. Grokking Simplicity: Taming Complex Software With Functional Thinking. Manning Publications, 2021. 550 с.
8. Трибюк В.О., Фант М.О. Стек технологій для реалізація платформи для косметичних колекцій та блогу. Тези доповідей VIII Всеукраїнської науково-технічної конференції «Комп'ютерні технології: інновації, проблеми, рішення», 02-03 грудня 2025 р. Житомир: Державний університет «Житомирська політехніка», 2025. С. 33–34.
9. React: Up And Running: Building Web Applications. O'Reilly Media, Incorporated, 2021. 222 с.
10. What is angular?. Angular. URL: <https://angular.dev/overview> (дата звернення: 11.02.2026).
11. Learning Typescript: Enhance Your Web Development Skills Using Type-Safe Javascript. O'Reilly Media, Incorporated, 2022.

12. Pattankar M., Hurbuns M. Mastering ASP.NET Web API. O'Reilly Media, Incorporated, 2017. 330 с.

13. Mahdhi M. Mastering ASP.NET Core 10: Web Applications Made Easy With The Biggest Update Yet. O'Reilly Media, Incorporated, 2025. 891 с.

14. What Is SQL Server?. Microsoft Learn: Build with answers in reach. URL: <https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server/> (дата звернення: 13.02.2026).

15. PostgreSQL: about. PostgreSQL: The world's most advanced open source database. URL: <https://www.postgresql.org/about/> (дата звернення: 13.06.2026).

16. About Blob (Object) Storage. Microsoft Learn: Build with answers in reach. URL: <https://learn.microsoft.com/uk-ua/azure/storage/blobs/storage-blobs-overview> (дата звернення: 15.04.2026).

17. Overview of Entity Framework Core. Microsoft Learn: Build with answers in reach. URL: <https://learn.microsoft.com/uk-ua/ef/core/> (date of access: 20.02.2026).

18. Miller R., Lerman J. Programming Entity Framework: Code First. O'Reilly Media, Incorporated, 2011.

19. Introduction To Semantic Kernel. Microsoft Learn: Build with answers in reach. URL: <https://learn.microsoft.com/en-us/semantic-kernel/overview/> (дата звернення: 20.05.2026).

20. Design Patterns Elements of Reusable Object-Oriented Software / G. Erich та ін. Longman (Pearson Education), 1995. 416 с.

21. Трибюк В.О., Фант М.О. Реалізація поведінкового шаблону Стратегія для обробки взаємодій користувачів з об'єктами системи. Тези доповідей XI Міжнародної науково-технічної конференції «Інформаційно-комп'ютерні технології», 02-03 квітня 2026 р. Житомир: Державний університет «Житомирська політехніка», 2026. С. 35-36.

22. Introduction to Identity on ASP.NET Core. Microsoft Learn: Build with answers in reach. URL: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity> (дата звернення: 11.03.2026).

23. Introduction | Slate. Slate. URL: <https://docs.slatejs.org/> (дата звернення: 13.05.2026).

24. Quartz.net. Quartz.NET. URL: <https://www.quartz-scheduler.net/> (дата звернення: 21.04.2026).

25. GIN indexes. PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/current/gin.html> (дата звернення: 08.05.2026).

26. Pg_trgm. PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/current/pgtrgm.html> (дата звернення: 24.05.2026).

27. Andersson J. C. Learning microsoft azure. O'Reilly Media, Incorporated, 2023. 480 с.

28. What is Azure Database for Postgresql?. Microsoft Learn: Build with answers in reach. URL: <https://learn.microsoft.com/en-us/azure/postgresql/overview> (дата звернення: 23.05.2026).

29. Overview of azure communication services email. Microsoft Learn: Build with answers in reach. URL: <https://learn.microsoft.com/uk-ua/azure/communication-services/concepts/email/email-overview> (дата звернення: 23.05.2026).

30. Introduction to SAS (shared access signature). Microsoft Learn: Build with answers in reach. URL: <https://learn.microsoft.com/en-us/shows/inside-azure-for-it/introduction-to-sas-shared-access-signature> (дата звернення: 14.05.2026).

31. Safe Storage of App Secrets in Development. Microsoft Learn: Build with answers in reach. URL: <https://learn.microsoft.com/en-us/aspnet/core/security/app-secrets?view=aspnetcore-10.0&tabs=windows,powershell> (дата звернення: 11.04.2026).

32. Normalizing. Slate. URL: <https://docs.slatejs.org/concepts/11-normalizing> (дата звернення: 13.05.2026).

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
						83
Змн.	Арк.	№ докум.	Підпис	Дата		

Діаграми використання



Рис. А.1. Діаграма використання «Взаємодія з колекцією»

Діаграми класів

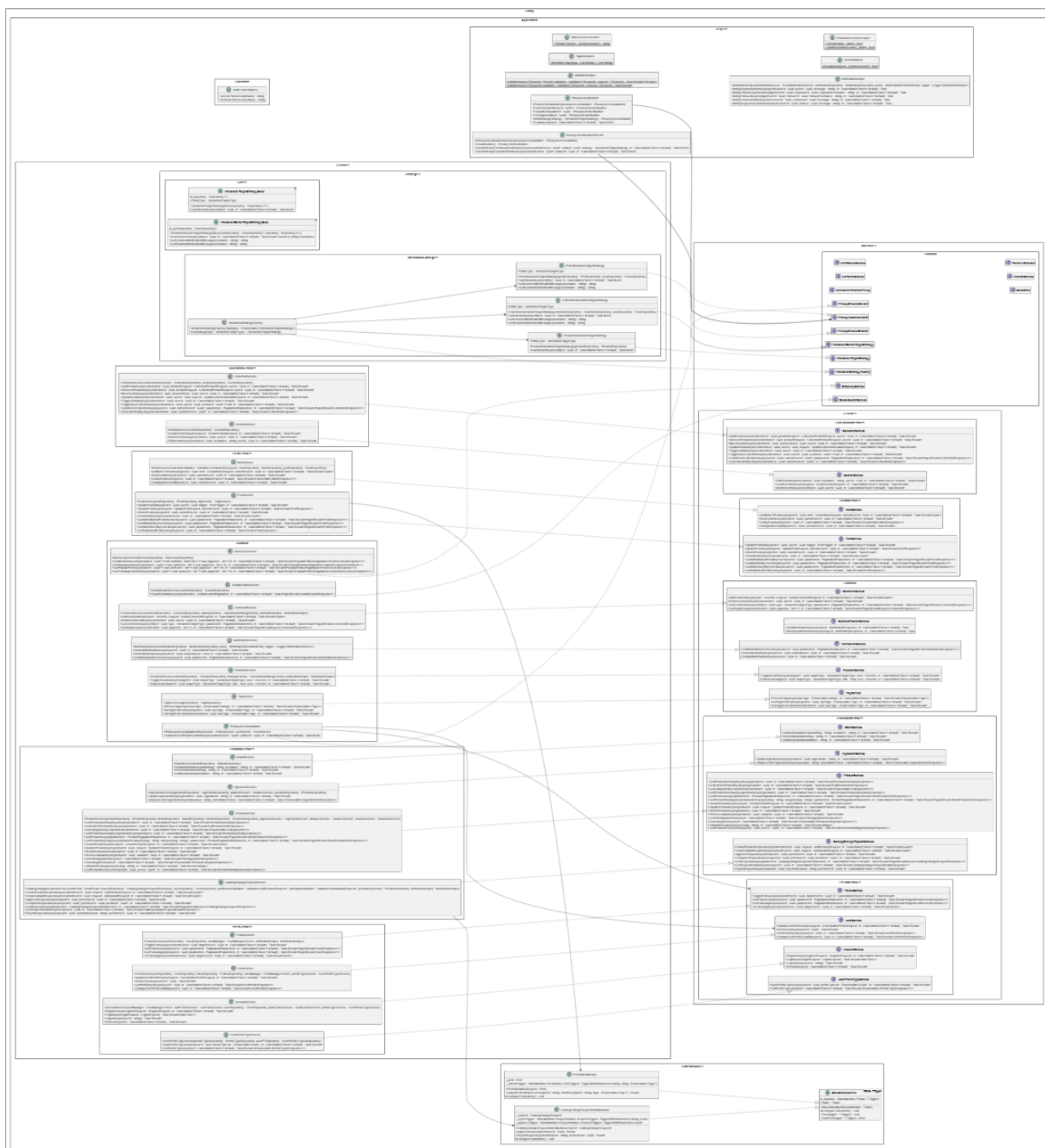


Рис. Б.1. Діаграма класів шар Application

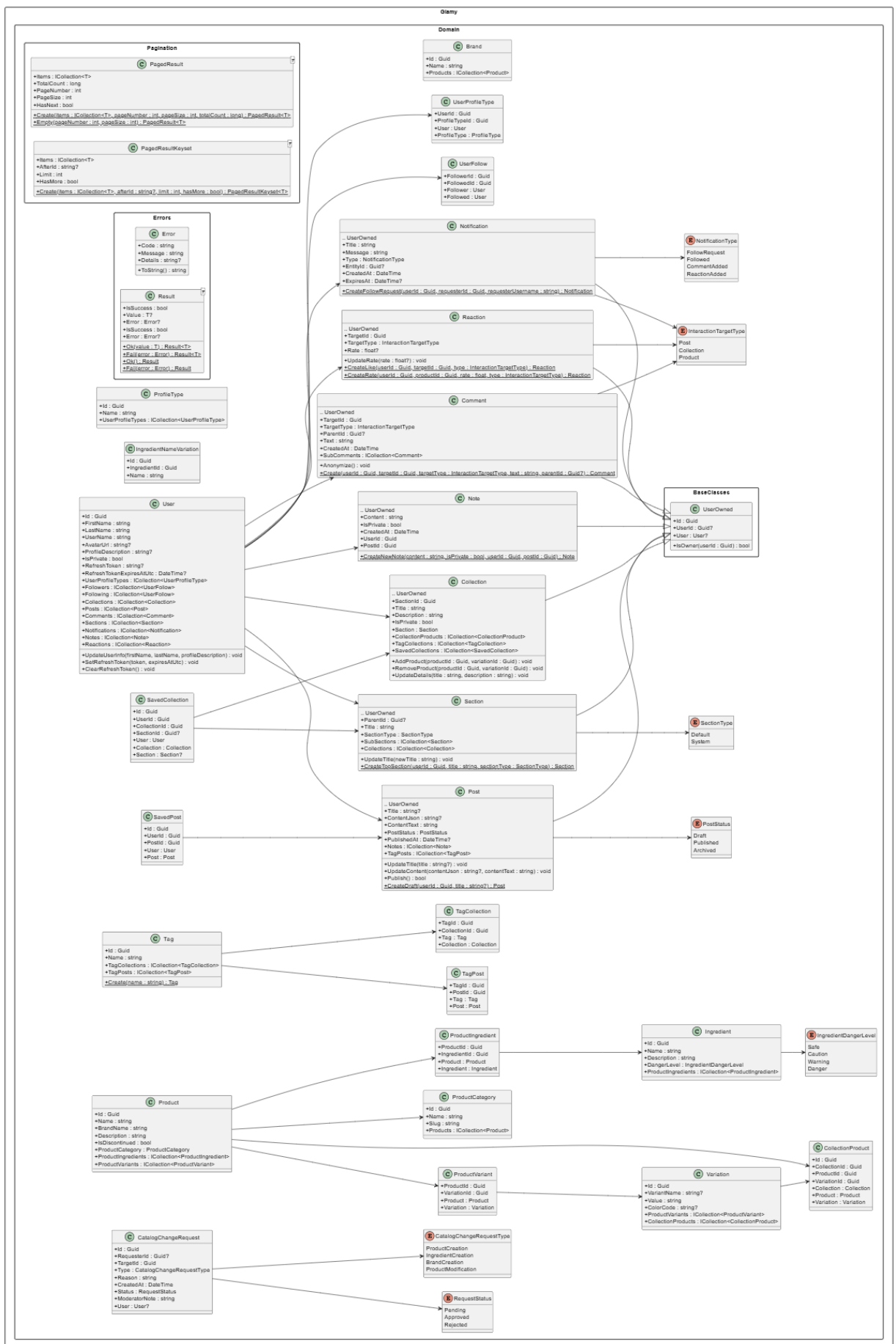


Рис. Б.2. Діаграма класів шар Domain

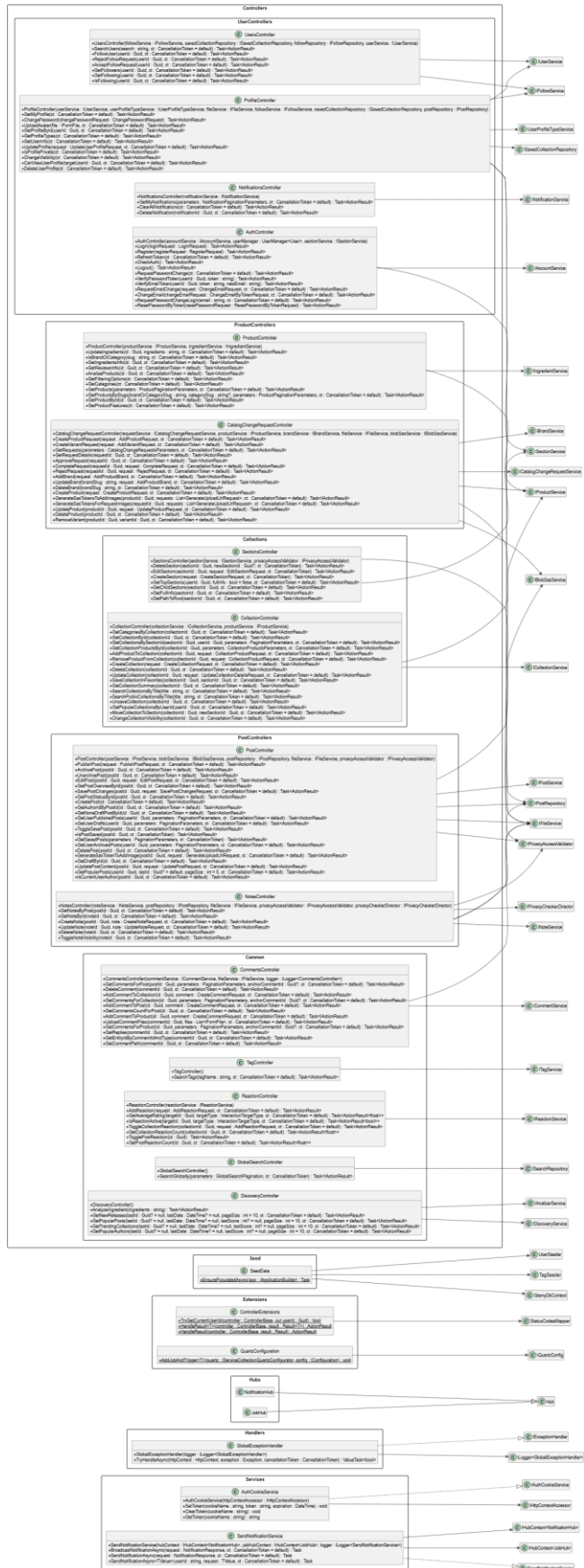


Рис. Б.4. Діаграма класів шар API

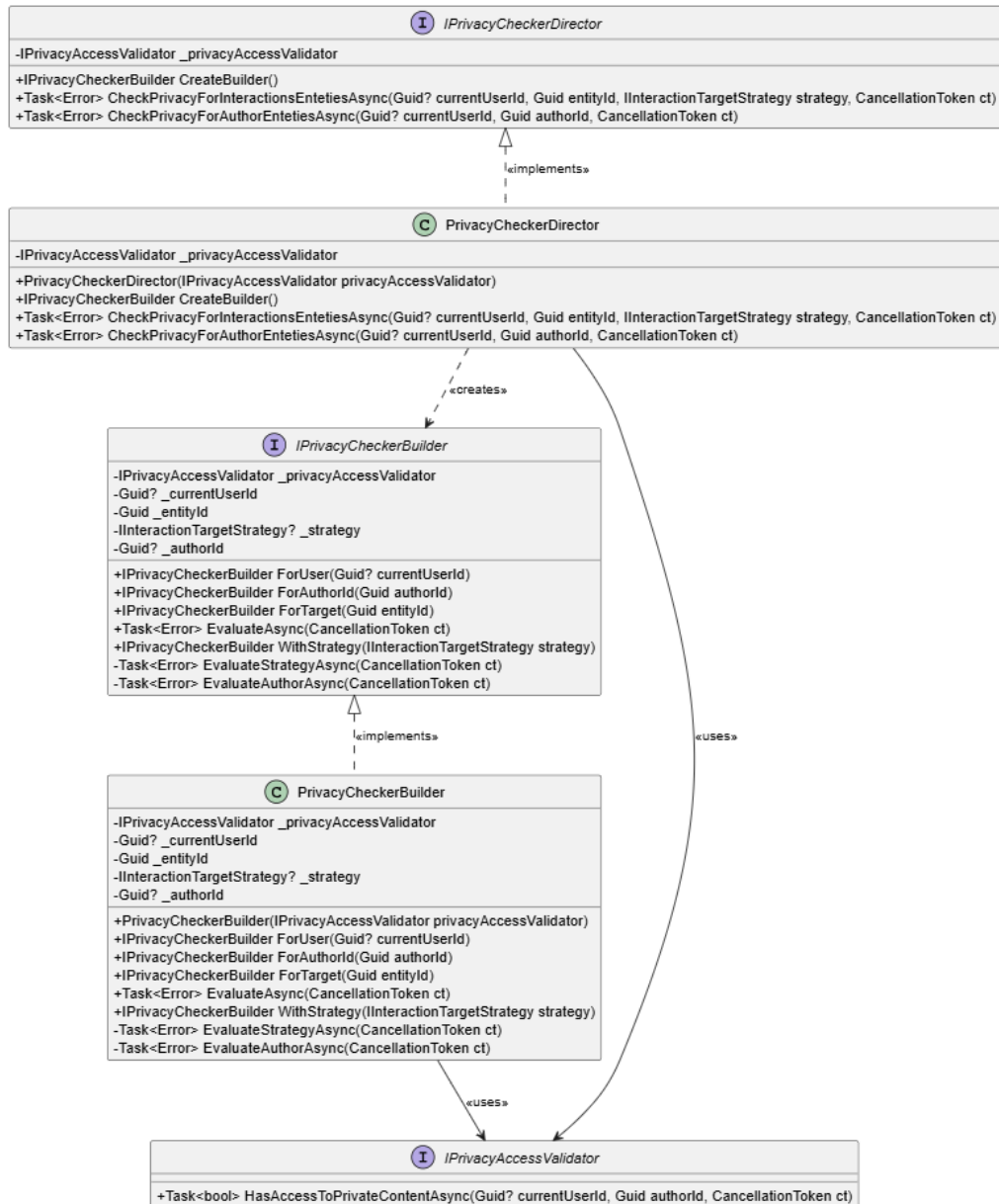


Рис. Б.5. Діаграма класів, реалізація патерну Будівельник

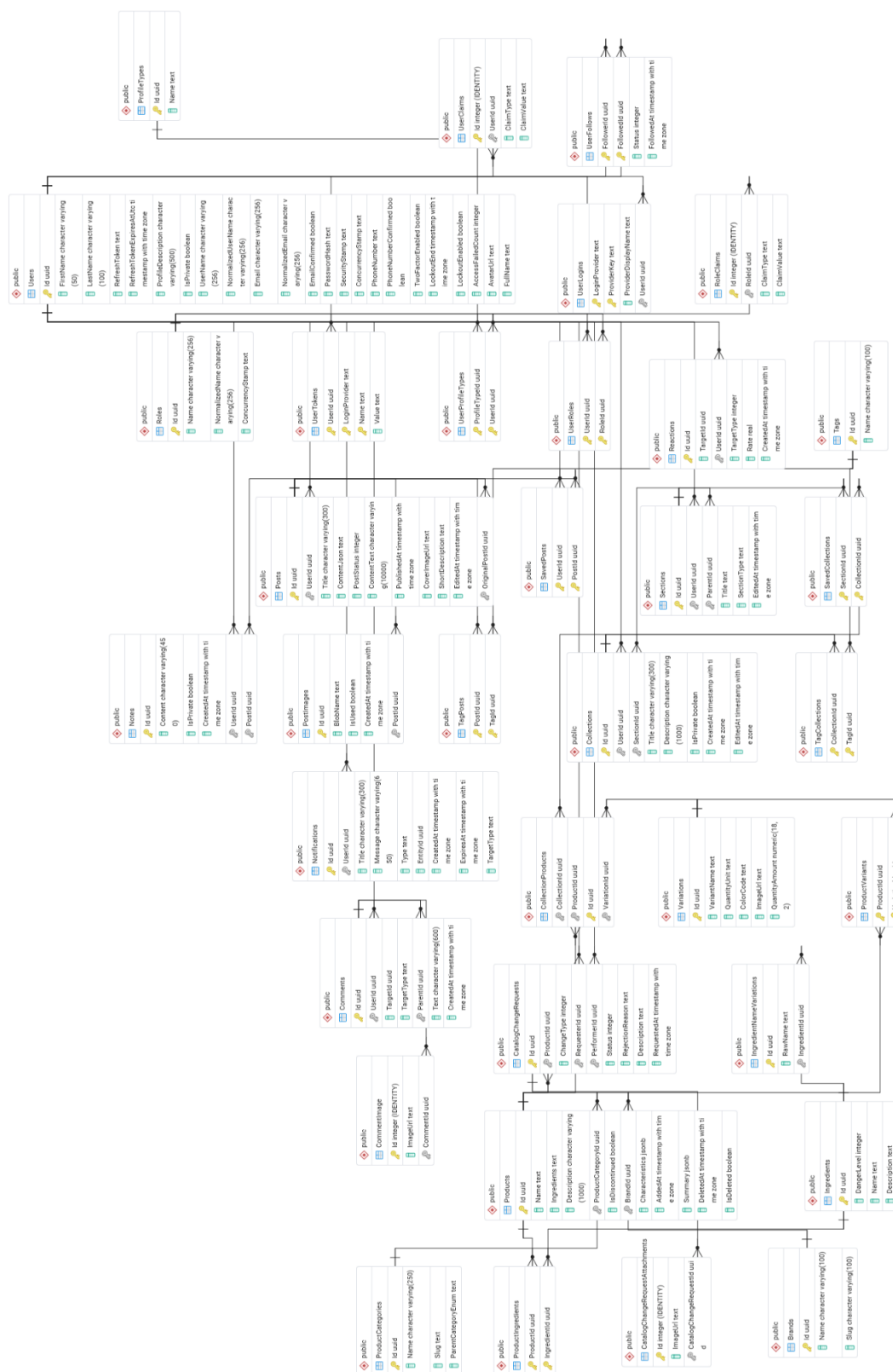


Рис. В.1 – Діаграма «сутність-зв'язок» (фізичний рівень) системи

Діаграми активності та послідовності

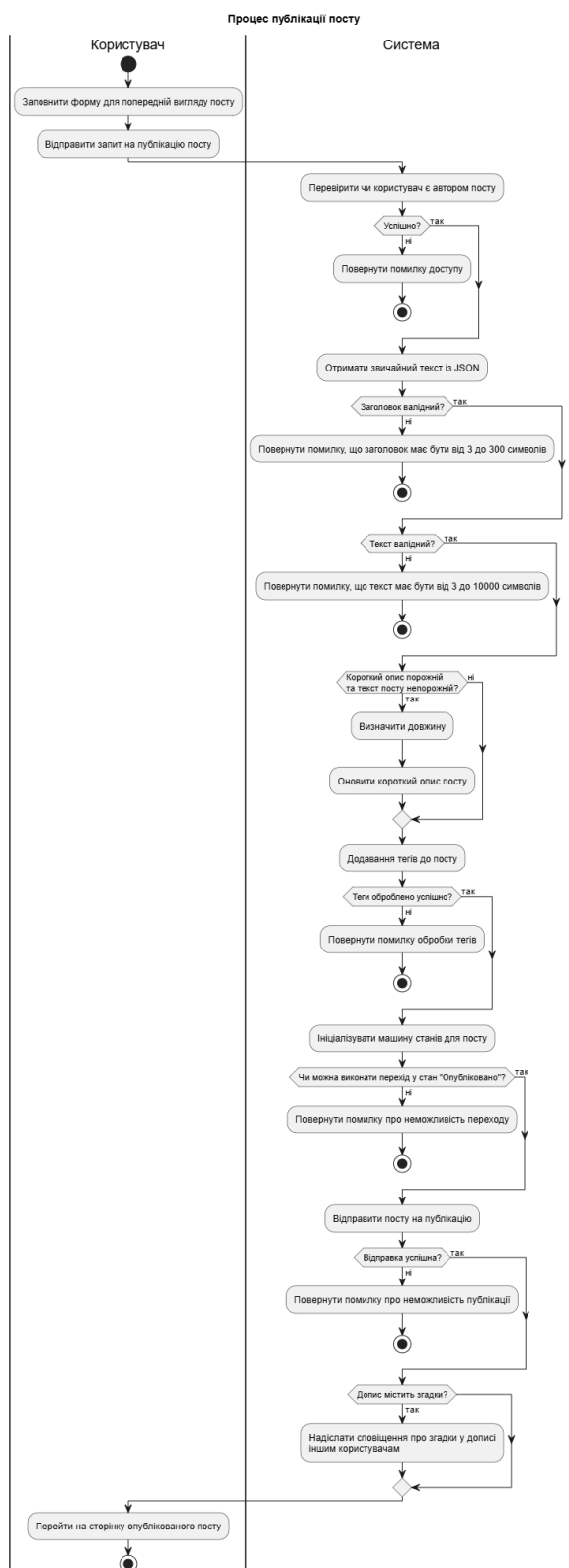


Рис. Д.1. Діаграма активності «Публікація посту»

