МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Відокремлений структурний підрозділ
«Тернопільський фаховий коледж Тернопільського національного технічного університету імені Івана Пулюя»

(повне найменування вищого навчального закладу)

Відділення телекомунікацій та електронних систем

(назва відділення)

Циклова комісія комп'ютерних наук

(повна назва циклової комісії)

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

фахового молодшого бакалавра

на тему: Розробка інтернет-магазину косметики «BeautyHeaven»

Виконав: студент (ка) 4 курсу, групи КН-421

спеціальності 122 «Комп'ютерні науки»

(шифр і назва спеціальності)

Івахов С.О.

(прізвище та ініціали)

Керівник Сербін В.С.

(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

м. Тернопіль – 2026

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ТЕРНОПІЛЬСЬКИЙ ФАХОВИЙ КОЛЕДЖ
ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ ІМЕНІ ІВАНА ПУЛЮЯ»

Відділення телекомунікацій та електронних систем
Циклова комісія комп'ютерних наук
Освітньо-професійний ступінь «фаховий молодший бакалавр»
Спеціальність 122 «Комп'ютерні науки»
Галузь знань 12 «Інформаційні технології»

ЗАТВЕРДЖУЮ
Голова циклової комісії комп'ютерних наук
_____ Галина МАРЦІЯШ
« _____» _____ 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Івахов Сергій Олександрович
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка інтернет-магазину косметики «BeautyHeaven»

керівник роботи: Сербін Володимир Сергійович,
(прізвище, ім'я, по батькові)
затверджені наказом вищого навчального закладу № 4/9-132 від 27.02.2026 р.

2. Строк подання студентом роботи: 19.06.2026 р.

3. Вихідні дані до роботи: мови програмування: Java, JavaScript; фреймворки: Spring Boot, Hibernate, Tailwind CSS; бібліотека React, Vite, стандарти IEEE 830-1998, IEEE 29148-2018, IEEE 29119, ГОСТ 34.602-89.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

Анотація

Вступ

1 Загальний розділ

1.1 Аналітичний огляд існуючих рішень

1.2 Технічне завдання

1.2.1 Найменування та область застосування

1.2.2 Призначення розробки

1.2.3 Вимоги до функціоналу web-сайту

1.2.4 Вимоги до програмної документації

1.2.5 Техніко-економічні показники

1.2.6 Стадії та етапи розробки

1.2.7 Порядок тестування та прийому

2 Розробка технічного та робочого проєкту

2.1 Розробка структури сайту і web-сторінок

2.2 Створення та верстка сторінок сайту

2.3 Розробка структури бази даних сайту

2.4 Програмування сайту

2.4.1 Написання клієнтської частини

2.4.2 Написання admin-частини

2.5 Тестування web- сайту

3 Спеціальний розділ

3.1 Інструкція з розміщення сайту в Інтернеті

3.2 Інструкція з обслуговування та наповнення сайту

3.3 Інструкція з популяризації та підтримки сайту

4 Економічний розділ

4.1 Визначення стадій технологічного процесу та загальної тривалості проведення НДР

4.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

4.3 Розрахунок матеріальних витрат

4.4 Розрахунок витрат на електроенергію

4.5 Розрахунок суми амортизаційних відрахувань

4.6 Обчислення накладних витрат

4.7 Складання кошторису витрат та визначення собівартості НДР

4.8 Розрахунок ціни НДР

4.9 Визначення економічної ефективності і терміну окупності капітальних вкладень

5 Охорона праці, техніка безпеки та екологічні вимоги

6 Висновки

Додаткові вказівки: виконання кваліфікаційної роботи із розробкою програмного продукту – вебсайту.

5. Перелік графічного матеріалу:

1. ER-діаграма бази даних.
2. Діаграма прецедентів (Use Case).
3. Діаграма послідовності (Sequence).
4. Таблиця техніко-економічних показників.

6. Консультанти розділів роботи

Розділ	**Ім'я та прізвище, посада консультанта**	**Підпис, дата**	
		завдання видав	**завдання прийняв**
Економічний розділ	Любов КАЛУШКА		
Охорона праці, техніка безпеки та екологічні вимоги	Генадій ГОРЯЧЕК		

1 КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту	Строк виконання етапів роботи	2 Примітка
1	Отримання і аналіз технічного завдання	20.03.2026	
2	Збір і узагальнення інформації по роботі	01.05.2026	
3	Написання першого розділу	15.05.2026	
4	Розробка технічного та робочого проєкту	29.05.2026	
5	Написання спеціального розділу	05.06.2026	
6	Розрахунок економічної частини	06.06.2026	
7	Написання розділу охорони праці	07.06.2026	
8	Виконання графічної частини	08.06.2026	
9	Оформлення роботи	09.06.2026	
10	Погодження нормоконтролю	.06.2026	
11	Попередній захист роботи	10.06.2026	
12	Захист роботи	24.06.2026	

7. Дата видачі завдання: 05 березня 2026 р.

Студент _____ Сергій ІВАХОВ
(підпис) (ім'я та прізвище)

Керівник роботи _____ Володимир СЕРБІН
(підпис) (ім'я та прізвище)

АНОТАЦІЯ

Тема кваліфікаційної роботи: Розробка інтернет-магазину косметики «BeautyHeaven».

Метою кваліфікаційної роботи є розробка повнофункціонального веб-додатку інтернет-магазину косметики на базі клієнт-серверної архітектури для забезпечення зручних онлайн-покупок та ефективного управління контентом.

Пояснювальна записка складається з 5 розділів.

У загальній частині описуються аналітичний огляд існуючих рішень у сфері електронної комерції та аналіз технічного завдання на розробку програмного продукту.

У другому розділі представлено процес створення програмного продукту, обґрунтування вибору технологій (React, Spring Boot, PostgreSQL), опис структури та методу організації бази даних, опис алгоритмів і інформаційних зв’язків, зовнішнє проєктування, тестування та налагодження програми.

В спеціальній частині описані процес розгортання (інсталяції) програмного продукту в хмарному середовищі, інструкція з експлуатації адміністративної частини сайту, а також інструкція з популяризації та підтримки.

Розрахунок вартості розробки та економічної ефективності приведено в економічній частині.

Основні питання охорони праці та техніки безпеки розглянуто в п’ятому розділі.

Обсяг пояснювальної записки 113 сторінок.

До складу кваліфікаційної роботи входить графічна частина, яка складається з діаграми прецедентів (Use Case), ER-діаграми реляційної бази даних та діаграми послідовності.

ABSTRACT

Theme of the qualification paper: Development of the cosmetics online store "Beauty Heaven".

The aim of the qualification paper is to develop a fully functional web application for a cosmetics online store based on a client-server architecture to provide convenient online shopping and effective content management.

The explanatory note consists of 5 chapters.

The general part describes an analytical overview of existing solutions in the field of e-commerce and an analysis of the technical specifications for the development of the software product.

The second chapter presents the process of creating the software product, the justification for the choice of technologies (React, Spring Boot, PostgreSQL), the description of the structure and method of database organization, the description of algorithms and information relationships, external design, testing, and debugging of the program.

The special part describes the process of deploying (installing) the software product in a cloud environment, the operation manual for the administrative part of the website, as well as instructions for promotion and technical support.

The calculation of development costs and economic efficiency is presented in the economic part.

The main issues of occupational health and safety are discussed in the fifth chapter.

The volume of the explanatory note is 113 pages.

The qualification paper includes a graphical part, which consists of a Use Case diagram, an Entity-Relationship (ER) database diagram, and a Sequence diagram.

ЗМІСТ

ВСТУП

Сьогодні сфера електронної комерції в Україні демонструє стабільне зростання, що робить розробку нових онлайн-майданчиків особливо актуальною. Покупці все частіше обирають швидкий шопінг зі смартфона замість довгих походів до торгових центрів. За статистикою аналітичного центру групи компаній EVO, у 2025 році обсяг онлайн-покупок українців досяг 256 мільярдів гривень. Це реальний приріст у 7% порівняно з минулим роком [1].

Змінилася і сама поведінка споживачів. Якщо раніше косметику намагалися купувати виключно наживо, щоб мати можливість протестувати текстуру чи аромат, то зараз ситуація інша. Користувачі здійснюють в середньому 17,5 замовлень на рік, а категорія «косметика та парфумерія» стабільно тримається в топі найпопулярніших товарів на українських маркетплейсах. Це прямий доказ того, що покупці готові витрачати гроші на б'юті-товари дистанційно, спираючись на опис та відгуки. Тому створення сучасного інтернет-магазину «BeautyHeaven» є логічним та технічно обґрунтованим кроком.

Актуальність розробки даного web-сайту підтверджується низкою вагомих переваг електронного формату торгівлі, які якісно змінюють підхід до покупок.

Зручність вибору та порівняння є особливо відчутною на тлі того, що в традиційних офлайн-магазинах площа вітрин завжди обмежена. До того ж, читати дрібний шрифт на баночках із кремом і намагатися запам'ятати склад, щоб порівняти його з іншим засобом - фізично незручно, а консультанти не завжди є вільними. Натомість вебсайт здатний агрегувати тисячі товарів у зручному каталозі. Користувач отримує детальні фільтри, за допомогою яких можна в кілька кліків відсіяти косметику з небажаними алергенами або знайти засіб під свій бюджет. Доступ до незалежних відгуків реальних покупців значно спрощує процес вибору та економить час.

Глибока персоналізація пропозицій успішно реалізується завдяки сучасним технологіям, за допомогою яких інтернет-магазин може працювати як розумний персональний асистент. Системи аналізують попередні дії користувача та історію

його покупок. Наприклад, якщо клієнт регулярно шукає засоби для сухої шкіри, сайт генеруватиме індивідуальні рекомендації саме з такими зволожуючими товарами. Це підвищує лояльність аудиторії та стимулює продажі.

Можливість впровадження інноваційного функціоналу зумовлена тим, що стандартні цифрові рішення вже не повністю задовольняють сучасних покупців, тому розробка власного сайту дозволяє реалізувати унікальні інструменти взаємодії. Цифровий формат дає змогу впровадити інтерактивну візуалізацію хімічного складу косметики, щоб клієнт наочно бачив дію компонентів. Крім того, онлайн-платформа дозволяє реалізувати функцію спільного кошика для кількох користувачів, що абсолютно неможливо відтворити у фізичному магазині.

Метою кваліфікаційної роботи є розробка web-сайту для функціонування інтернет-магазину косметики з клієнтською та адміністративною частинами.

Для досягнення поставленої мети необхідно виконати такі завдання:

1. Провести аналітичний огляд існуючих web-рішень у сфері електронної комерції та сформувати технічне завдання
2. Спроєктувати структуру сайту та бази даних.
3. Здійснити програмну реалізацію клієнтської та адміністративної частин web-сайту, виконати верстку сторінок.
4. Провести тестування розробленого web-сайту.
5. Розробити інструкції з розміщення, наповнення та підтримки сайту в мережі Інтернет.
6. Розрахувати техніко-економічні показники проєкту.
7. Визначити вимоги до охорони праці, техніки безпеки та екології при розробці та експлуатації програмного продукту.

Галуззю можливого використання розробленого програмного рішення є підприємства роздрібної та гуртової торгівлі, малий та середній бізнес у сфері б’юті-індустрії (продаж косметики, парфумерії, засобів догляду). Запропонований web-сайт може використовуватися як готовий шаблон для швидкого запуску онлайн-продажів або як база для подальшого масштабування торгової мережі.

1. ЗАГАЛЬНИЙ РОЗДІЛ

1.1. Аналітичний огляд існуючих рішень

Сучасний ринок електронної комерції у сегменті косметики та засобів догляду за собою переживає етап стрімкої трансформації. Спостерігається глобальний тренд на свідоме споживання: сучасні покупці стають більш обізнаними, ретельно вивчають хімічний склад продуктів (INCI)[2], уникають шкідливих компонентів та підбирають догляд відповідно до специфічних потреб шкіри. Проте, незважаючи на зміну патернів споживчої поведінки, більшість існуючих торговельних майданчиків не надають користувачам відповідних технологічних інструментів. Інформація про інгредієнти здебільшого подається у вигляді неструктурованого тексту, що створює високе когнітивне навантаження[3] та змушує клієнтів використовувати сторонні ресурси для перевірки безпечності складників.

Міжнародна номенклатура косметичних інгредієнтів (International Nomenclature of Cosmetic Ingredients, INCI) була створена з метою уніфікації маркування продукції та забезпечення прозорості для споживачів. Відповідно до міжнародних стандартів, виробники зобов'язані вказувати всі інгредієнти у порядку зменшення їх масової частки в рецептурі. Проте, на практиці ця система створює суттєвий бар'єр для пересічного користувача.

Більшість хімічних сполук позначаються складними латинськими термінами або англомовними абревіатурами. З точки зору когнітивної психології, зіткнення з неструктурованим масивом специфічних термінів викликає у користувача ефект когнітивного перевантаження. Замість того, щоб отримати інформацію про корисність чи безпечність продукту, клієнт змушений самостійно здійснювати пошук інформації по кожному компоненту. Це особливо критично для людей з чутливою шкірою, алергіями, або тих, хто дотримується етичних принципів (наприклад, обирає виключно cruelty-free або vegan косметику).

Іншим важливим аспектом, який ігнорується більшістю сучасних платформ,

є соціальна природа шопінгу. Історично електронна комерція розвивалася за моделлю ізольованих транзакцій (Web 2.0), де один користувач взаємодіє з системою віч-на-віч. Однак сучасні тренди вказують на перехід до соціальної комерції. Користувачі прагнуть обговорювати покупки в реальному часі, радитися з дерматологами або друзями безпосередньо під час вибору товару. Відсутність інтегрованих механізмів, таких як синхронізовані кошики або сесії спільного перегляду, змушує клієнтів використовувати сторонні месенджери, копіювати посилання та вручну підраховувати загальну вартість замовлення. Це розриває безперервність користувацького досвіду та збільшує відсоток покинутих кошиків.

Для визначення архітектурних недоліків та обґрунтування напрямку розробки веб-додатка «BeautyHeaven», було проведено аналіз трьох популярних існуючих рішень: Makeup [4], Notino [5] та Eva [6].

1.1.1 Аналіз платформи Makeup

Makeup є одним із лідерів національного ринку beauty-комерції. Під час дослідження даної системи встановлено, що вона володіє потужною підсистемою каталогізації, яка здатна обробляти сотні тисяч товарних позицій. Серед сильних сторін виявлено зручну багаторівневу фільтрацію та наявність базового аналізатора складу, який надає коротку інформацію про окремі компоненти.

Однак, детальний розгляд функціоналу виявив суттєві обмеження. Наявний аналізатор працює виключно в межах одного товару і не здатен порівнювати два різні засоби на предмет хімічної сумісності. Наприклад, користувач не зможе дізнатися про потенційний конфлікт активних речовин, якщо він додасть до кошика сироватку з саліциновою кислотою та сироватку з ретинолом, що в реальних умовах може призвести до подразнення шкіри[7]. З точки зору процесу оформлення замовлення, система використовує стандартну модель одиночного кошика. Можливість шерингу сесії або одночасного додавання товарів різними користувачами в єдине замовлення технічно не реалізована.

1.1.2 Аналіз платформи Notino.ua

Система електронної торгівлі Notino.ua відзначається сучасним користувацьким інтерфейсом (UI/UX) та потужною контентною складовою. Платформа активно використовує поради від експертів у власному блозі, що сприяє утриманню аудиторії.

Незважаючи на це, технічна реалізація сторінки товару має значні прогалини у роботі з даними. Встановлено повну відсутність функції аналізатора складу: інгредієнти виводяться суцільним, нерозміченим текстом без жодної інтерактивності.

Відповідно, користувач позбавлений інструментарію для порівняння двох товарів на предмет синергії або конфліктів інгредієнтів. Проблема «ізольованого шопінгу» тут також залишається невирішеною: якщо два користувачі вирішують зробити спільне замовлення на платформі Notino, вони змушені проходити складний позасистемний шлях: один з користувачів має надіслати посилання на товари через зовнішній месенджер, а інший - вручну шукати ці товари на сайті, перевіряти наявність обраного об'єму та додавати їх у свій локальний кошик. Будь-які зміни в наявності товару або зміна рішення одного з учасників вимагають повторного циклу комунікації за межами платформи.

1.1.3 Аналіз платформи Eva

Платформа Eva представляє собою високоякісний програмний продукт із продуманим візуальним дизайном, наявністю детальних описів товарів, б'юті-порад та інструкцій щодо способів використання косметики.

Проте, аналіз функціональних можливостей сайту засвідчив, що інтерфейс аналізу інгредієнтів повністю відсутній. Це призводить до розриву користувацького досвіду: покупець змушений копіювати незрозумілі латинські назви компонентів та шукати їх розшифровку у зовнішніх пошукових системах. Система не встановлює реляційних зв'язків між хімічними елементами у базі даних.

Крім того, платформа орієнтована виключно на індивідуального покупця, ігноруючи потребу аудиторії у спільній взаємодії під час вибору товарів.

1.1.4 Порівняльна характеристика рішень та обґрунтування розробки

На основі проведеного аналітичного огляду виявлено, що провідні гравці ринку успішно вирішують базові завдання електронної комерції, такі як каталогізація, обробка платежів та багаторівнева фільтрація. Проте вони не задовольняють зростаючі потреби сучасного споживача у глибокій аналітиці хімічного складу косметики та соціальній взаємодії.

Вивчення фахової літератури та методів структурування даних показує, що більшість існуючих систем використовують найпростішу реляційну або текстову модель для збереження складу товару. Інгредієнти зберігаються як єдиний рядок типу String або плоский масив Array. Така логічна схема унеможливлює складний аналіз. З математичної точки зору, найбільш ефективним методом вирішення задачі аналізу сумісності є використання хімічної теорії графів, яка, згідно з сучасними дослідженнями, надає дієву основу для математичного опису того, як молекулярні взаємодії та зв'язки впливають на властивості системи через структуру вузлів і ребер [8]. Спираючись на ці принципи, логічну модель складу косметичного засобу доцільно представити у вигляді неорієнтованого зваженого графа G = (V, E), де:

1. V (вершини) - це множина інгредієнтів у продукті (v_1, v_2, , v_n), що виступають математичними аналогами компонентів суміші.

2. E (ребра) - це множина взаємозв'язків між інгредієнтами (синергія, нейтральність або конфлікт), що допомагає аналізувати структуру їхньої сумісності.

Вага ребра у такому графі може позначати рівень сумісності компонентів. Існуючі торговельні майданчики не реалізують подібних математичних моделей на рівні своїх баз даних, що робить неможливим побудову інтерактивних переглядачів.

Також під час аналізу систем провідних конкурентів (таких як Makeup,

Notino та Eva) встановлено, що всі вони використовують парадигму ізольованого керування станом [9]. Інформаційна схема роботи таких платформ передбачає жорстку прив'язку сесії замовлення до унікального ідентифікатора одного клієнта (Session ID або JWT-токена) та локального сховища браузера. У цій моделі взаємодія відбувається виключно за синхронним протоколом HTTP за схемою «запит-відповідь» (Request-Response). Дослідження сучасних архітектурних патернів підтверджують, що традиційне сесійне керування станом кошика, яке застосовується в цих системах, не пристосоване для багатокористувацьких дій і є вразливим до проблеми стану гонки (Race Condition) при одночасних зверненнях [10]. Якщо два користувачі спробують одночасно авторизуватися під одним акаунтом для спільного наповнення кошика, неминуче виникає конфлікт паралельних запитів, що призводить до перезапису стану кошика та втрати даних одного з клієнтів. З метою систематизації отриманих під час аналізу даних та виявлених архітектурних недоліків, було сформовано порівняльну таблицю ключових функціональних характеристик існуючих систем та запропонованого проєкту «BeautyHeaven», яку наведено в таблиці 1.1.

Таблиця 1.1 - Порівняльний аналіз систем аналогів

Критерії порівняння	Makeup.com.ua	Notino.ua	Eva	Проєкт «BeautyHeaven»
Наявність каталогу товарів	Наявна	Наявна	Наявна	Наявна
Текстовий склад товарів	Наявний (базовий аналіз)	Наявний (суцільний текст)	Наявний (суцільний текст)	Наявний (розмічений)
Інтерактивний аналіз сумісності інгредієнтів	Відсутній	Відсутній	Відсутній	Наявний (візуалізація зв'язків)
Спільний кошик	Відсутній	Відсутній	Відсутній	Наявний

Доведено, що розробка інтернет-магазину косметики «BeautyHeaven» є технічно доцільним та інноваційним завданням. Проєктована система вирішуватиме реальні проблеми користувачів шляхом впровадження унікальних функціональних модулів:

1. Інтерактивного переглядача складу, який трансформує незрозумілий текстовий опис у візуальні графи (інтерактивні бульбашки зі зв'язками), що наочно демонструють синергію або конфлікти інгредієнтів.

2. Алгоритму автоматизованого порівняння двох товарів на предмет їх сумісності в одному циклі догляду.

3. Модуля «Спільного кошика» із застосуванням технологій двостороннього зв'язку (WebSockets) для забезпечення real-time синхронізації, що дозволить кільком користувачам одночасно та безперешкодно формувати єдине замовлення з різних акаунтів [11].

Таким чином, реалізація зазначених архітектурних та інтерфейсних рішень дозволить створити конкурентоспроможний програмний продукт, що позбавлений критичних недоліків існуючих платформ-гігантів.

1.2. Технічне завдання

1.2.1. Найменування та область застосування

Повне найменування програмного виробу: «BeautyHeaven».

Галузь застосування програмного виробу – сфера електронної комерції (e-commerce) у сегменті краси та здоров'я, орієнтована на бізнес-модель B2C (Business-to-Consumer). Об'єктом, у якому передбачається використання розробки, є процес роздрібної онлайн-торгівлі косметичними засобами, що потребує глибокої аналітики хімічного складу товарів та інструментів для колективної взаємодії користувачів під час здійснення покупок.

Цільовою аудиторією продукту є як пересічні споживачі, що шукають базовий догляд, так і вимогливі клієнти (наприклад, люди з чутливою шкірою,

алергіями або специфічними дерматологічними потребами), для яких критично важливо розуміти хімічну взаємодію компонентів перед покупкою. З точки зору бізнесу, розроблена платформа може бути впроваджена як готове програмне рішення для діючих підприємств роздрібної торгівлі, нішевих б'юті-бутиків або стартапів, які прагнуть вийти на ринок з інноваційним підходом до обслуговування клієнтів. Програмний виріб розрахований на функціонування у хмарному середовищі з кросплатформним доступом через сучасні веббраузери.

1.2.2 Призначення розробки

Експлуатаційне призначення: забезпечення користувачів (клієнтів) зручним, надійним та інформативним інструментом для вибору, аналізу та придбання косметичних засобів у режимі онлайн, а також забезпечення адміністраторів системи ефективними засобами управління контентом магазину.

Досягнення поставленої експлуатаційної мети забезпечується шляхом реалізації наступних засобів:

1. багатокористувацької системи реєстрації та авторизації;
2. інтерактивного каталогу товарів з багаторівневою фільтрацією;
3. модуля візуалізації хімічного складу товару (побудова інтерактивних графів інгредієнтів);
4. системи спільного кошика для колективного формування замовлень у режимі реального часу;
5. панелі адміністратора для ручного керування даними про товари, бренди, категорії, інгредієнти та їхні взаємозв'язки.

Найважливішим функціоналом розроблюваного вебсайту, який якісно виділяє його серед існуючих конкурентів на ринку, є інтеграція модуля графової аналітики складу та системи «Спільного кошика». Аналітичний модуль вирішує ключовий біль споживача: він трансформує складний неструктурований текст стандарту INCI у візуальну карту інгредієнтів, дозволяючи наочно досліджувати синергію або конфлікти активних речовин косметики. Водночас імплементація

технології WebSockets для спільного кошика створює унікальний соціальний досвід шопінгу (Social Commerce). Цей функціонал дозволяє користувачам генерувати посилання-запрошення та спільно, з різних пристроїв і без перезавантаження сторінок, наповнювати єдине замовлення, бачити дії один одного та спільно застосовувати промокоди.

Впровадження такого комплексного функціоналу дозволить не лише значно знизити відсоток повернень товарів через небажані алергічні реакції, але й підвищити загальний рівень довіри покупців до платформи, стимулюючи зростання середнього чека за рахунок гейміфікації та спрощення процесу колективних покупок.

1.2.3 Вимоги до функціоналу web-сайту

Програмний виріб обробляє кілька потоків вхідних даних, джерелом яких виступають як звичайні покупці, так і адміністратор системи. Перелік основних вхідних даних наведено у таблиці 1.2

Таблиця 1.2 – Перелік і опис вхідних повідомлень

Назва вхідного повідомлення	**Форма подання**	**Періодичність надходження**	**Джерело даних / Користувач**
1	**2**	**3**	**4**
Облікові дані користувача (ім'я, прізвище, email, пароль, телефон, адреса)	Екранна web-форма	При реєстрації або оновленні профілю	Покупець
Інформація про контент (товари, бренди, категорії, інгредієнти, зв'язки)	Екранна web-форма	За необхідності (оновлення асортименту)	Адміністратор

Продовження таблиці 1.2

1	2	3	4
Події управління кошиком (додавання/видалення товару)	WebSocket-повідомлення (JSON)	У режимі реального часу під час сесії	Покупець
Пошукові та фільтраційні запити	Екранна web-форма	За ініціативою користувача	Покупець

Для кожного виду вхідних даних система формує відповідні вихідні повідомлення для користувачів:

1. Каталог та деталі товару: подаються у вигляді екранної форми (UI) на запит покупця.

2. Інтерактивний граф інгредієнтів: подається у вигляді візуалізованої схеми (на базі бібліотеки react-force-graph-2d) на сторінці товару за запитом покупця.

3. Стан спільного кошика: подається у вигляді списку обраних товарів та загальної суми, що оновлюється на екрані (UI) в режимі реального часу для всіх учасників спільної сесії.

4. Аналітична інформація та списки замовлень: подаються у вигляді табличних даних в адміністративній панелі для керуючого персоналу.

Програмний виріб повинен виконувати наступні основні процедури:

1. Процедура авторизації та керування сесіями: обробляє вхідні облікові дані, валідує їх та генерує вихідний JWT (JSON Web Token) для подальшої автентифікації запитів.

2. Процедура управління контентом: отримує вхідні дані від адміністратора (інформацію про склад косметики, ціни, наявність) та виконує їх запис у реляційну базу даних PostgreSQL.

3. Процедура візуалізації складу: отримує з БД інформацію про інгредієнти конкретного товару та зв'язки між ними (вхідні дані з сервера), обробляє їх на

стороні клієнта і генерує вихідну візуалізацію у вигляді графа.

4. Процедура синхронізації спільного кошика: обробляє двосторонні з'єднання за протоколом WebSocket. При надходженні команди на додавання товару від одного користувача, сервер миттєво розсилає оновлений стан кошика (вихідні дані) всім учасникам поточної кімнати покупок.

Для наочного відображення складу системи, ролей користувачів та розподілу бізнес-логіки за функціональними вимогами було розроблено відповідну модель прецедентів. Діаграма прецедентів (Use Case) спроєктованого веб-сайту зображена в Додатку А.

Обмеження*:* архітектура системи обмежується необхідністю надійного підключення до мережі Інтернет для забезпечення роботи WebSockets.

Жорсткі часові обмеження (системи реального часу критичного рівня) до даного програмного виробу не ставляться. Проте, з метою забезпечення позитивного користувацького досвіду (UX), встановлюються наступні ліміти:

1. час завантаження та рендерингу сторінок каталогу не повинен перевищувати 3 секунд за умови стабільного інтернет-з'єднання;

2. затримка синхронізації стану спільного кошика між клієнтами не повинна перевищувати 1 секунду.

З метою забезпечення надійного функціонування системи передбачено:

1. Ієрархічна структура комплексу: чіткий поділ на клієнтську (Frontend) та серверну (Backend) частини для ізоляції помилок.

2. Захист від несанкціонованого доступу: реалізовано шляхом використання JWT токенів для доступу до захищених API-маршрутів та хешування паролів у базі даних.

3. Контроль вводу даних: дворівнева валідація даних (на стороні клієнта за допомогою React та на стороні сервера засобами Java Spring) для перевірки форматів email, довжини паролів та коректності числових значень.

4. Обробка виняткових ситуацій: перехоплення помилок (наприклад, «Користувача не знайдено», «Товар видалено») з видачею користувачу дружніх повідомлень без зупинки роботи додатку.

5. Збереження даних: використання транзакцій у PostgreSQL для забезпечення цілісності інформації під час фінансових операцій чи формування замовлень.

Умови експлуатації:

1. Режим роботи програми: багатокористувацький, цілодобовий онлайн-режим (24/7).

2. Взаємодія користувача: здійснюється через графічний інтерфейс користувача (GUI) з можливістю введення даних через клавіатуру та маніпулятор «миша» або сенсорні екрани мобільних пристроїв.

3. Програмна сумісність: клієнтська частина оптимізована для роботи у сучасних веб-браузерах (Google Chrome, Safari, Mozilla Firefox, Microsoft Edge) з обов'язковою підтримкою виконання JavaScript та технології WebSockets.

4. Інструментальне середовище: клієнтська частина розроблена з використанням бібліотеки React та фреймворку Tailwind CSS; серверна частина функціонує на базі Java Spring; для зберігання інформації використовується СКБД PostgreSQL.

Дотримання зазначених вимог щодо умов експлуатації та функціональних характеристик гарантує стабільну роботу платформи, високу швидкість обробки даних та забезпечує позитивний користувацький досвід при взаємодії з системою.

1.2.4 Вимоги до програмної документації

Склад програмної документації визначається складністю програмного виробу, його архітектурою та потребами користувачів і персоналу, що обслуговує систему. Комплект документації повинен включати:

1. Пояснювальну записку до кваліфікаційної роботи.
2. Текст програми (лістинг вихідного коду), винесений у додатки.
3. Інструкцію з розміщення сайту в Інтернеті.
4. Інструкцію з обслуговування та наповнення сайту (керівництво адміністратора).

5. Інструкцію з популяризації та підтримки сайту.

6. ER-діаграма бази даних

До розробленого програмного коду висуваються наступні спеціальні вимоги щодо документування:

1. кожний програмний модуль або процедурний блок (як на стороні клієнта у React, так і на сервері у Spring) повинен мати початковий блок коментарів, який містить: призначення модуля, склад вхідних та вихідних даних, обмеження та умови використання, а також дату введення в дію;

2. коментарі у коді повинні бути стислими, чіткими та однозначними;

3. згідно з принципами структурного програмування, кожний модуль повинен мати суворо одну точку входу та одну точку виходу.

Наявність повної, структурованої та якісно оформленої програмної документації забезпечить легкість подальшого супроводу продукту, а також дозволить швидко залучати нових фахівців до процесу масштабування інтернет-магазину.

1.2.5 Техніко-економічні показники

Розробка сучасного комерційного веб-додатку вимагає чіткого планування та раціонального розподілу ресурсів. Техніко-економічні показники є фундаментальною базою для оцінки ефективності процесу створення програмного продукту. Для реалізації інтернет-магазину косметики «Beauty Heaven» було залучено комплекс трудових та апаратних (машинних) ресурсів, які забезпечили виконання повного циклу розробки: від етапу збору вимог до фінального розгортання проєкту в хмарному середовищі.

Реалізація мікросервісної архітектури проєкту (розділення на клієнтську та серверну частини) вимагала залучення спеціалістів різних профілів. Загальна трудомісткість розробки склала 158 людино-годин. Для забезпечення високої якості коду та дотримання термінів виконання, ці трудовитрати були розподілені між чотирма основними ролями:

1. Керівник проєкту (Project Manager): Витрачено 25 людино-годин. Цей час був спрямований на планування архітектури системи, аналіз ринку, проєктування структури реляційної бази даних, розробку технічного завдання, а також підготовку супровідної проєктної документації.

2. Frontend-розробник: Витрачено 60 людино-годин. Трудовий ресурс був використаний для створення візуальної концепції сайту (UI/UX дизайн) та безпосереднього написання клієнтської частини на базі бібліотеки React. Сюди входить верстка компонентів, налаштування маршрутизації (React Router) та інтеграція інтерфейсу з серверним API.

3. Backend-розробник: Витрачено 58 людино-годин. Основний обсяг часу був виділений на розробку серверної логіки з використанням фреймворку Java Spring Boot. Роботи включали налаштування безпеки (Spring Security), реалізацію бізнес-логіки обробки товарів та замовлень, підключення до бази даних PostgreSQL, а також налаштування безперервної інтеграції та розгортання на платформі Render.

4. QA-інженер: Витрачено 15 людино-годин. Ресурс витрачено на проведення ручного та автоматизованого тестування, перевірку валідації електронних форм, виявлення програмних дефектів (багів) та перевірку адаптивності веб-додатку на різних пристроях.

Згідно з нормативними документами, стандартний робочий місяць одного фахівця при 40-годинному робочому тижні становить в середньому 160-168 робочих годин. Оскільки сумарна трудомісткість проєкту склала 158 годин, загальний обсяг витрачених трудових ресурсів еквівалентний 1 людино-місяцю чистого робочого часу. Однак, зважаючи на технологічні паузи, необхідність узгодження рішень та паралельне виконання завдань різними спеціалістами, фактичний календарний термін реалізації проєкту від стадії планування до введення в експлуатацію склав 2 місяці.

Окрім людського капіталу, розробка програмного забезпечення вимагає залучення обчислювальних потужностей. Загальний обсяг виділеного машинного часу повністю корелює з часом роботи спеціалістів за персональними

комп'ютерами та становить 158 годин машинного часу.

Цей машинний час був витрачений на наступні операції:

1. Компіляція та збірка вихідного коду.
2. Локальний запуск та тестування Docker-контейнерів для симуляції ізольованого серверного середовища.
3. Робота з системами управління базами даних (генерація SQL-запитів, виконання міграцій через інструмент Flyway).
4. Синхронізація локального коду з віддаленими репозиторіями (GitHub) та взаємодія з хмарними сервісами (Vercel, Render) в режимі реального часу.

Для забезпечення безперебійної роботи та високої швидкості компіляції використовувалася сучасна робоча станція (персональний комп'ютер) балансовою вартістю 50 000 грн. Витрати електроенергії на забезпечення машинного часу склали 1 259 грн при середньому споживанні обладнання 0,5 кВт/год.

Додатково, використання сучасних хмарних рішень (PaaS-платформ) дозволило суттєво знизити капітальні витрати на етапі запуску, оскільки виключило необхідність придбання та обслуговування власних фізичних серверів. Це створює сприятливі умови для подальшого масштабування проєкту, адже витрати на інфраструктуру зростатимуть виключно пропорційно до збільшення реального клієнтського навантаження.

Оптимальний розподіл 158 годин трудового та машинного часу дозволив мінімізувати фінансові витрати на реалізацію проєкту. Загальна собівартість розробки склала 90 328 грн. Завдяки високій затребуваності рішень у сфері електронної комерції (e-commerce), проєкт має високий рівень рентабельності. При плановому грошовому потоці в 50 584 грн на рік, термін повної окупності витрачених на розробку ресурсів становить 2,1 року. Це свідчить про те, що обсяг виділених трудових та машинних ресурсів є економічно обґрунтованим, а сам проєкт — ефективним та життєздатним.

1.2.6 Стадії та етапи розробки

Процес розробки програмного виробу базується на ітеративній моделі життєвого циклу і поділяється на наступні основні стадії:

1. Аналіз та підготовка (Етап 1): вивчення предметної області, аналіз існуючих рішень конкурентів, збір вимог, формування та затвердження технічного завдання.

2. Проєктування (Етап 2): розробка архітектури клієнт-серверного додатка, проєктування логічної та фізичної структури бази даних PostgreSQL, створення UI/UX макетів веб-сторінок.

3. Програмування (Етап 3): розробка серверної частини (Backend) з використанням Java Spring, налаштування REST API:

   – реалізація клієнтської частини (Frontend) на базі React та Tailwind CSS;
   – імплементація специфічних модулів: інтеграція бібліотеки react-force-graph-2d для побудови графів та налаштування WebSockets для спільного кошика.

4. Тестування та налагодження (Етап 4): проведення ручного функціонального тестування та інтеграційного тестування серверної частини (API) за допомогою Postman, перевірка кросбраузерності, виявлення та усунення помилок.

5. Впровадження та супровід (Етап 5): розгортання веб-додатка на сервері (хостингу), написання експлуатаційної документації та оформлення пояснювальної записки.

Окрім основних стадій, життєвий цикл розробки постійно підтримувався практиками безперервної інтеграції. На етапі програмування активно використовувалася система контролю версій Git. Кожен новий складний функціонал розроблявся в окремих гілках (branches), після чого відбувалося злиття (merge) з основною кодовою базою. Такий підхід, заснований на принципах методології Agile, дозволив гнучко реагувати на зміни вимог та своєчасно усувати архітектурні конфлікти.

Чітке дотримання визначеної послідовності стадій життєвого циклу дозволяє оптимізувати процес розробки програмного забезпечення, мінімізувати ризики виникнення архітектурних помилок та гарантувати своєчасне завершення проєкту.

1.2.7 Порядок контролю та прийому

Працездатність програмного виробу перевіряється в умовах, максимально наближених до реальної експлуатації. Для проведення контролю необхідна наявність персонального комп'ютера з доступом до мережі Інтернет та встановленим сучасним веб-браузером.

Контроль та прийом здійснюється шляхом виконання контрольного прикладу (набору ручних функціональних тестів), що включає наступні сценарії:

1. Тестування доступу та безпеки: демонстрація реєстрації нового користувача, валідація введених даних, авторизація в системі та отримання JWT-токена.

2. Тестування інтерактивного каталогу: перехід до картки косметичного товару, запуск модуля аналізу складу та перевірка коректності відображення візуального графа інгредієнтів і зв'язків між ними.

3. Тестування спільного кошика (Multi-user Cart): імітація паралельної роботи двох користувачів з різних пристроїв/браузерів. Відкриття спільної сесії кошика, додавання та видалення товарів одним користувачем із перевіркою миттєвої (real-time) синхронізації стану кошика на екрані іншого користувача.

4. Тестування адміністративної частини: вхід під обліковим записом адміністратора, демонстрація процесу додавання нового товару та створення зв'язків між інгредієнтами в базі даних.

Успішне проходження всіх перелічених сценаріїв контрольного прикладу є фактичною підставою для визнання програмного виробу таким, що повністю відповідає початковим вимогам технічного завдання та є готовим до фінального розгортання і подальшої експлуатації.

2. РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ

2.1 Розробка структури сайту і web-сторінок

2.1.1 Призначення ресурсу та розробка візуальної концепції

Відповідно до затвердженого технічного завдання (ТЗ), розроблюваний веб-додаток «BeautyHeaven» за своїм призначенням є платформою електронної комерції, що поєднує в собі функціонал класичного інтернет-магазину косметики, інформаційно-аналітичної системи (для аналізу складів) та інструментів соціальної взаємодії (спільний кошик).

Робота над інтерфейсом (UI/UX) розпочалася зі створення візуальної концепції головної сторінки та затвердження фірмового стилю. Кольорова палітра проєкту була підібрана з урахуванням сучасних принципів психології в UI/UX дизайні, що безпосередньо впливає на когнітивне сприйняття бренду та рівень конверсії [12]**.** Основу дизайну складають відтінки рожевого, червоний (для акцентування уваги на критичних діях, таких як додавання до кошика або помилках) та білий колір, який використовується як «повітря» для забезпечення чистоти інтерфейсу та зниження когнітивного навантаження при читанні великих масивів даних.

Для візуалізації цих архітектурних рішень на початковому етапі проєктування було створено низькодеталізований прототип (Wireframe) за допомогою графічного редактора Figma. Побудова такого структурного макета без використання реального контенту дозволила абстрагуватися від дрібниць та зосередити увагу виключно на композиції, ергономіці розташування ключових елементів і правильному розподілі кольорових акцентів. Розроблений прототип ілюструє базову модульну сітку сторінки, розміщення глобальної навігації, робочої області та відображає загальну стилістичну спрямованість системи (див. рис. 2.1).

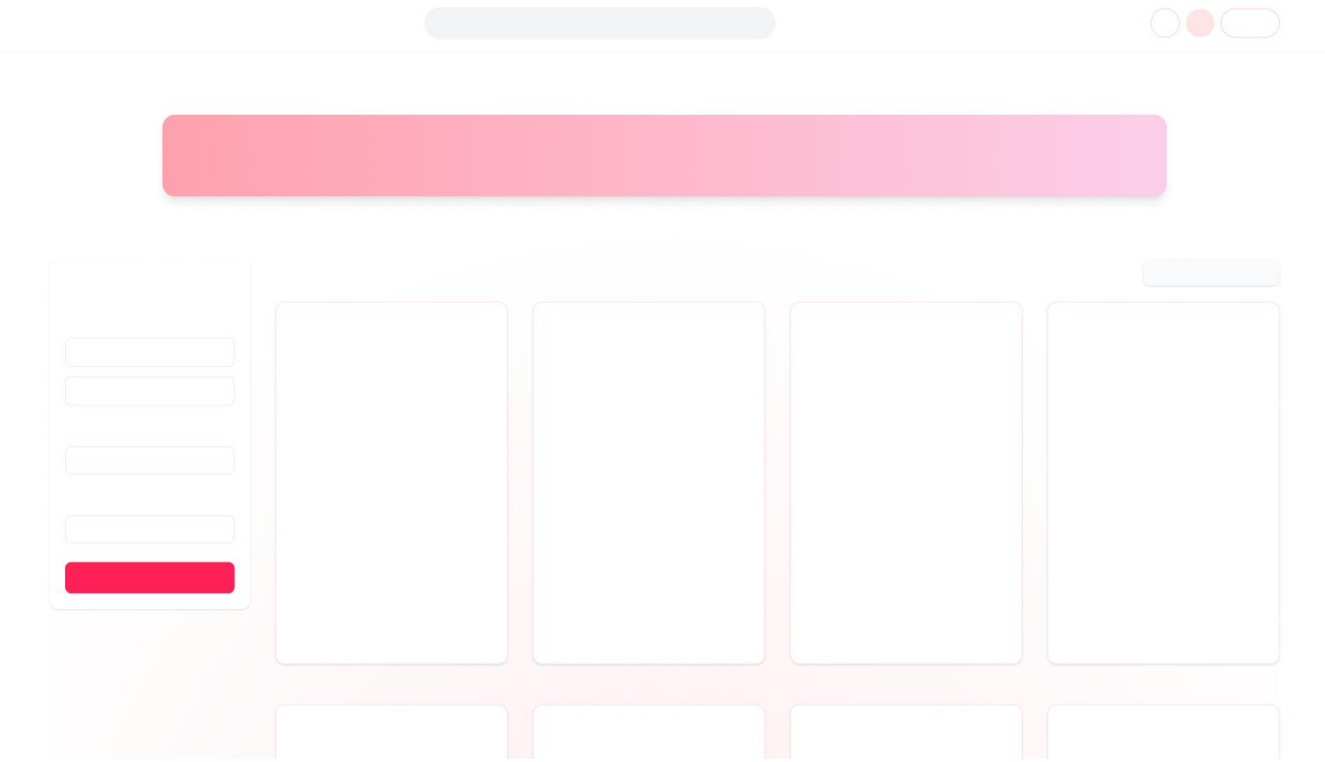


Рисунок 2.1 – Низькодеталізований прототип інтерфейсу веб-додатка «BeautyHeaven»

Такий підхід до раннього візуального прототипування дозволив затвердити базову структуру інтерфейсу із замовником та уникнути архітектурних і логічних помилок на етапах детального дизайну та безпосередньої програмної верстки додатка.

2.1.2 Інформаційна архітектура та карта сайту

Для забезпечення логічної навігації та швидкого доступу до функціоналу було спроєктовано інформаційну архітектуру платформи згідно з сучасними стандартами проєктування інтерфейсів електронної комерції [13]. Карта сайту має ієрархічну структуру та поділяється на дві основні зони доступу: клієнтську та адміністративну.

До основних web-сторінок платформи належать:

1. Головна сторінка (Home Page): вітрина магазину з маркетинговими

банерами та каталогом.

2. Сторінка товару (Product Details Page): детальний опис косметичного засобу, його ціна, характеристики та інтерактивний модуль аналізу інгредієнтів.

3. Профіль користувача (User Dashboard): сторінка керування персональними даними, адресами доставки та історією замовлень.

4. Звичайний кошик (Standard Cart): інтерфейс індивідуального оформлення списку покупок.

5. Спільний кошик (Shared/Multi-user Cart): спеціалізована сторінка-кімната для колективних покупок із синхронізацією даних.

6. Оформлення замовлення (Checkout): сторінка введення платіжних реквізитів та вибору способу доставки.

7. Адміністративна панель (Admin Panel): закритий інтерфейс для керування товарним асортиментом, базою інгредієнтів та зв'язками між ними.

2.1.3 Розробка глобальної навігації

Глобальна навігація системи спроєктована таким чином, щоб користувач міг звернутися до найважливіших функцій з будь-якої сторінки сайту. Верхній колонтитул (Header) має горизонтальну компоновку та розташований у верхній частині екрана. Зліва направо у Header розміщено:

1. Логотип «BeautyHeaven» (виконує функцію повернення на головну сторінку);

2. Рядок глобального пошуку товарів за назвою;

3. Іконка кошика (з індикатором кількості доданих товарів);

4. Іконка профілю користувача (для входу/реєстрації або переходу в особистий кабінет);

5. Кнопка «Вийти» для завершення сесії авторизованого користувача.

2.1.4 Проєктування інтерфейсу головної сторінки (каталогу)

Дизайн головної сторінки базується на сучасних патернах електронної комерції та правилах візуальної ієрархії (зокрема патернах сканування екрана), які довели свою ефективність у дослідженнях зорової уваги та користувацького досвіду [14]. Верхню частину займає маркетинговий банер, що інформує про новинки або акції. Робоча область поділена на дві колонки.

Зліва розташована панель фільтрації, яка дозволяє звузити вибір косметики за брендом, призначенням, ціновим діапазоном чи типом шкіри. У правій частині робочої зони знаходиться панель сортування (наприклад, від найдешевших до найдорожчих). Основну площу займає сітка карток товарів (див. рис. 2.2).

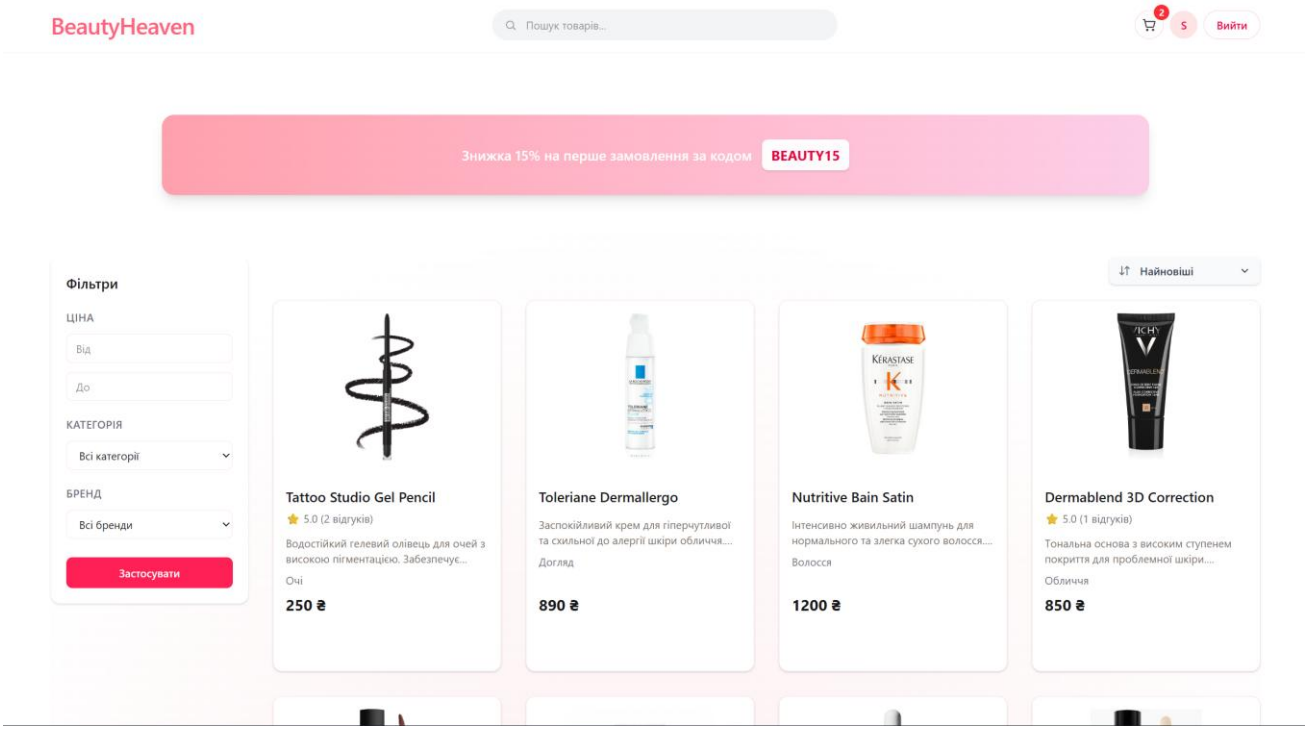


Рисунок 2.2 – Інтерфейс головної сторінки веб-додатка «BeautyHeaven»

Запропонована композиція головної сторінки забезпечує швидкий доступ до асортименту та створює інтуїтивно зрозумілу систему навігації, що є критично важливим фактором для утримання уваги потенційного покупця.

2.1.5 Проєктування сторінки товару та модуля аналізу складу

Особлива увага під час проєктування інтерфейсу приділялася сторінці детальної інформації про товар. Верхній блок містить стандартну інформацію: зображення товару, назву, ціну та кнопку «Додати до кошика».

У нижній частині сторінки розміщено інноваційний блок аналізу продукту. Тут склад косметичного засобу виводиться не суцільним текстом, а у вигляді інтерактивного списку інгредієнтів. При натисканні на конкретний інгредієнт система виводить спливаюче інформаційне вікно з детальним описом його властивостей. Також у цьому блоці спроєктовано кнопку «Інтерактивна карта інгредієнтів». Натискання на неї ініціює перехід до візуального інтерфейсу на базі технології Canvas, де користувач бачить граф хімічних компонентів (бульбашки та лінії зв'язків), що значно покращує рівень сприйняття складної інформації. (див. рис. 2.3).

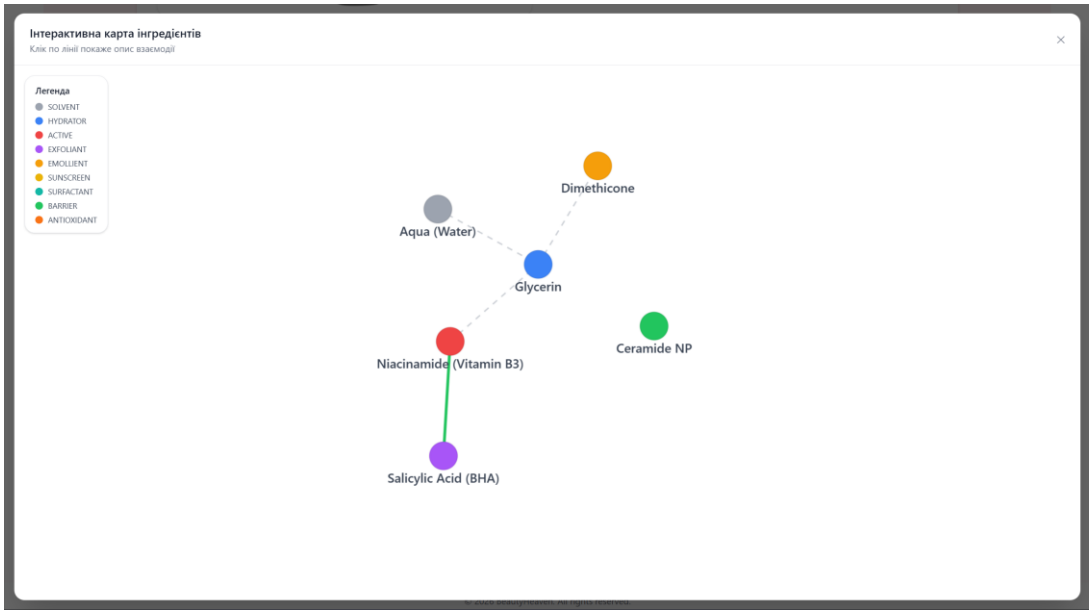


Рисунок 2.3 – Інтерфейс сторінки товару та інтерактивної карти інгредієнтів

Впровадження такого візуального компонента перетворює складну для сприйняття хімічну термінологію (INCI) на зручний аналітичний інструмент, що якісно виділяє розроблений продукт серед конкурентів та допомагає клієнту

зробити безпечний вибір.

2.1.6 Проєктування інтерфейсу підсистем оформлення замовлення

Система передбачає два роздільні інтерфейси для керування покупками. Звичайний кошик має традиційну структуру: перелік товарів, можливість зміни їх кількості, загальна сума та кнопка переходу до оплати (Checkout).

Зовсім інший UI/UX підхід застосовано для сторінки спільного кошика. Оскільки цей модуль передбачає одночасну роботу кількох осіб, його інтерфейс доповнено двома ключовими елементами:

1. Блок генерації запрошення: містить кнопку «Поділитися посиланням», яка генерує унікальний URL (ідентифікатор кімнати) для відправки іншим користувачам.

2. Панель «Live» (Учасники): спеціальний візуальний блок, де в режимі реального часу відображаються піктограми (аватарки та імена) користувачів, які наразі підключені до даної сесії кошика через WebSocket-з'єднання.

Це рішення перетворює звичайний процес покупки на соціальну взаємодію, дозволяючи учасникам бачити, хто саме та які товари додає до спільного замовлення (див. рис. 2.4).

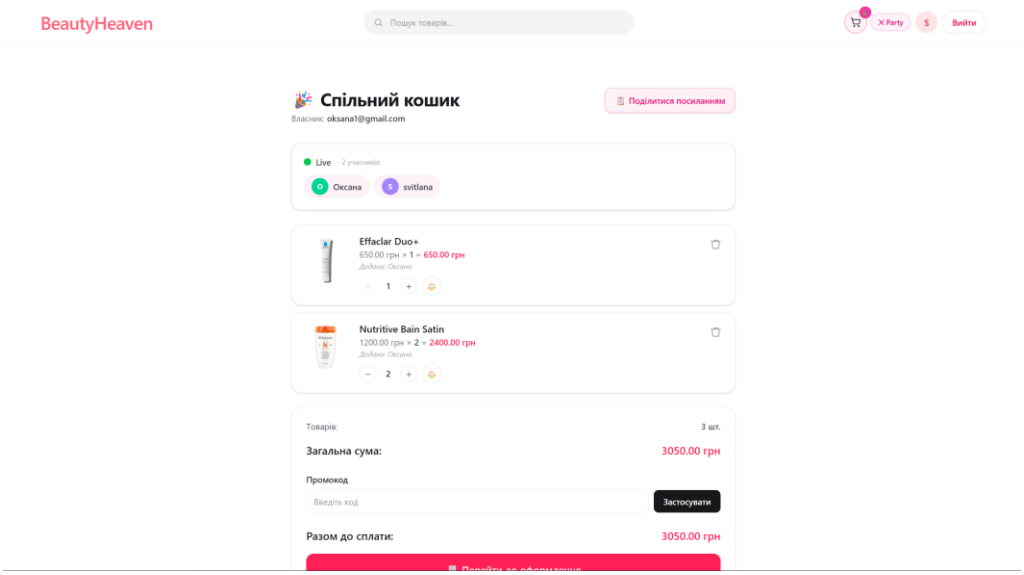


Рисунок 2.4 – Інтерфейс спільного кошика (Multi-user Cart) з панеллю «Live»

Подібна архітектура інтерфейсу інтегрує принципи соціальних мереж у класичний процес електронної комерції, перетворюючи ізольовану покупку на спільний інтерактивний досвід у режимі реального часу.

2.1.7 Проєктування адміністративної панелі

Дизайн адміністративної панелі (Back-office) спроєктовано за принципом максимальної утилітарності. Інтерфейс складається з навігаційного меню для швидкого перемикання між сутностями: «Аналітика» «Товари», «Бренди», «Категорії», «Інгредієнти» «Промокоди» та «Замовлення». Основна робоча область містить таблиці даних з підтримкою операцій CRUD (Створення, Читання, Оновлення, Видалення) та екранні форми для ручного введення інформації про косметичні засоби (див. рис. 2.5).

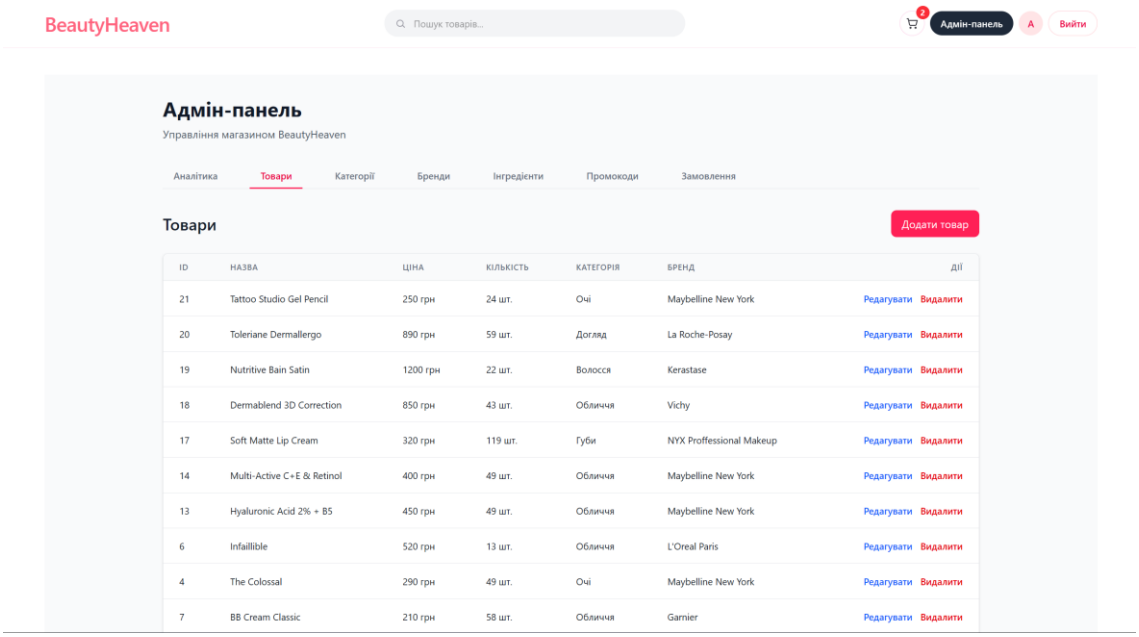


Рисунок 2.5 – Інтерфейс адміністративної панелі управління контентом сайту

Таким чином, розроблена структура сайту та макети web-сторінок повністю відповідають вимогам технічного завдання, забезпечуючи зручну навігацію та інтуїтивно зрозумілу взаємодію як для клієнтів, так і для адміністраторів платформи.

2.2 Створення та верстка сторінок сайту

2.2.1 Вибір середовища розробки та інструментарію

Процес створення програмного коду, верстки та стилізації інтерфейсу користувача здійснювався у сучасному інтегрованому середовищі розробки (IDE) Visual Studio Code (VS Code). Вибір даного редактора коду обґрунтований його високою продуктивністю, широкою підтримкою екосистеми JavaScript/TypeScript та можливістю розширення функціоналу за рахунок плагінів.

Для забезпечення безпомилкової та швидкої верстки із застосуванням обраного CSS-фреймворку, у середовище розробки було інтегровано офіційне розширення Tailwind CSS IntelliSense. Цей інструмент забезпечив автоматичне доповнення коду (autocomplete) для класів стилів, підсвічування синтаксису, а також відображення інформації про згенеровані CSS-правила безпосередньо під час написання коду, що значно пришвидшило процес перенесення візуальних макетів у програмний код.

2.2.2 Компонентний підхід та розробка JSX-шаблонів

Відповідно до архітектури сучасних веб-додатків, розробка інтерфейсу системи «BeautyHeaven» здійснювалася не за допомогою класичних статичних HTML-сторінок, а на базі бібліотеки React, яка є одним із галузевих стандартів для створення реактивних користувацьких інтерфейсів[15]. Результатом етапу верстки стали зверстані JSX-шаблони (JavaScript XML) - спеціальний синтаксис, що дозволяє писати HTML-подібний код безпосередньо всередині JavaScript-функцій.

Процес верстки базувався на принципах компонентно-орієнтованого програмування. Замість створення монолітних сторінок, інтерфейс був декомпозиваний на дрібні, незалежні та перевикористовувані UI-компоненти (наприклад, Header, ProductCard, CartItem, Button). Це дозволило уникнути дублювання коду та забезпечити легкість подальшого масштабування системи.

Для забезпечення навігації між зверстаними сторінками (Головна, Сторінка товару, Кошик тощо) без фізичного перезавантаження сторінки у браузері, була застосована концепція Single Page Application (SPA), що дозволяє динамічно оновлювати контент, забезпечувати високу швидкість рендерингу та знижувати навантаження на сервер[16]. Маршрутизація клієнтської частини реалізована за допомогою бібліотеки react-router-dom. Вона дозволила зв'язати URL-адреси браузера з відповідними React-компонентами сторінок, створюючи ілюзію класичного переходу між web-сторінками.

2.2.3 Стилізація за допомогою методології Utility-First

Для стилізації розроблених JSX-шаблонів було використано фреймворк Tailwind CSS, який пропагує методологію Utility-First (використання атомарних класів-утиліт). Згідно з офіційною документацією технології, такий підхід значно прискорює процес розробки та дозволяє підтримувати консистентність дизайну[17].

Процес верстки полягав у додаванні готових класів до атрибута className у JSX-тегах. Наприклад, для створення структури та задання кольору (відтінки рожевого, червоного та білого) використовувалися стандартні класи фреймворку: flex, grid, bg-pink-500, text-white, p-4, rounded-lg тощо. Завдяки багатій стандартній палітрі Tailwind CSS, візуальна концепція проєкту була реалізована без необхідності внесення змін до конфігураційного файлу tailwind.config.js.

Такий підхід дозволив:

1. Уникнути проблеми конфлікту імен CSS-класів;
2. Зменшити розмір кінцевого CSS-бандлу (оскільки Tailwind компілює лише ті класи, які фактично використані у розмітці);
3. Значно пришвидшити процес позиціонування елементів за допомогою технологій Flexbox та CSS Grid.

2.2.4 Скриптування елементів дизайну та управління станами

На етапі верстки також було здійснено скриптування інтерактивних елементів дизайну, як того вимагає технічне завдання. Оскільки проєкт базується на функціональних компонентах React, управління поведінкою інтерфейсу (відкриття модальних вікон, перемикання вкладок, показ інформації про інгредієнти) було реалізовано за допомогою вбудованих хуків (React Hooks), які були стандартизовані у специфікації React версії 16.8 [18].

Основними інструментами скриптування UI стали:

1. Хук useState: використовувався для збереження локального стану компонентів. Наприклад, при верстці інтерактивного блоку складу косметики, useState контролював булеве значення (true/false), яке визначало, чи відображати спливаюче інформаційне вікно з детальним описом конкретного інгредієнта при натисканні на нього користувачем. Також цей хук відповідав за керування станом видимості бічного меню або модальних вікон авторизації.

2. Хук useEffect: застосовувався для обробки побічних ефектів (Side Effects) на етапі життєвого циклу компонента. Під час верстки динамічних сторінок (наприклад, каталогу товарів), цей хук використовувався для первинного рендерингу та підготовки інтерфейсу до отримання даних.

Крім базових локальних станів, для синхронізації лічильника товарів у кошику між різними незалежними компонентами, активно застосовувався хук useContext. Такий підхід дозволив успішно уникнути надмірного каскадного передавання даних та значно спростив підтримку архітектури клієнтського коду.

Завдяки інтеграції JavaScript-логіки безпосередньо у процес верстки, розроблені сторінки стали реактивними: вони миттєво реагують на дії користувача, змінюючи свій візуальний стан без необхідності звернення до сервера за новими HTML-документами.

Підсумком етапу розробки стали повністю зверстані, стилізовані та скриптовані компоненти веб-додатка, які готові до інтеграції з серверною частиною (Backend) та базою даних проєкту.

2.3 Розробка структури бази даних сайту

2.3.1 Концептуальне та логічне проєктування бази даних

Продумана структура бази даних є фундаментальною основою для створення ефективної, масштабованої та відмовостійкої інформаційної системи. Процес проєктування бази даних веб-додатка «BeautyHeaven» являв собою комплексний процес відображення опису предметної області (електронної комерції та хімічного аналізу косметики) у схему внутрішньої реляційної моделі даних.

Враховуючи вимоги до транзакційної цілісності (ACID), підтримки складних зв'язків та необхідності швидкої обробки аналітичних запитів, згідно з сучасними рекомендаціями щодо проєктування масштабованих систем[19], як систему керування базами даних (СКБД) було обрано PostgreSQL.

На етапі логічного проєктування предметну область було декомпозировано на сутності (об'єкти), визначено їхні атрибути (властивості) та встановлено зв'язки між ними. Для усунення надмірності даних та забезпечення їхньої цілісності, структуру бази даних було приведено до третьої нормальної форми (3NF), що є академічним та індустріальним стандартом для реляційних баз даних електронної комерції [20].

2.3.2 Визначення таблиць та їхніх атрибутів за модулями

Для зручності опису та адміністрування, всі 13 розроблених таблиць бази даних можна логічно розділити на чотири функціональні модулі: модуль управління користувачами, модуль каталогу, модуль графової аналітики складу та модуль оформлення замовлень (спільний кошик).

Модуль управління користувачами відповідає за збереження облікових записів та авторизацію. Головною таблицею цього модуля є таблиця users, яка призначена для зберігання інформації про зареєстрованих клієнтів та адміністраторів. Вона містить наступні атрибути: первинний ключ id типу BIGINT,

ім'я first_name та прізвище last_name (типу VARCHAR), унікальну електронну адресу email (VARCHAR, UNIQUE), зашифрований пароль password (VARCHAR), контактний телефон phone (VARCHAR) та адресу доставки address (TEXT). Для розділення прав доступу покупців та адміністраторів використовується поле role (VARCHAR), а дата реєстрації фіксується у полі created_at (TIMESTAMP).

Модуль каталогу та контенту забезпечує збереження інформації про товарний асортимент інтернет-магазину. Довідник категорій товарів (наприклад, "Догляд за обличчям", "Макіяж") реалізовано за допомогою таблиці categories, яка складається з ідентифікатора id (BIGINT, PK), назви name (VARCHAR) та опису description (TEXT). Інформація про виробників косметики зберігається у таблиці-довіднику brands, що включає первинний ключ id (BIGINT, PK), назву бренду name (VARCHAR), опис description (TEXT) та країну походження country (VARCHAR).

Центральною таблицею каталогу є products, до якої входять первинний ключ id (BIGINT, PK), назва товару name (VARCHAR), його опис description (TEXT), вартість price (DECIMAL) та наявна кількість на складі stock_quantity (INTEGER). Зв'язок із категоріями та брендами забезпечується через зовнішні ключі category_id (BIGINT, FK) та brand_id (BIGINT, FK), а час створення запису фіксується у created_at (TIMESTAMP). Крім того, модуль містить таблицю відгуків користувачів reviews, що складається з первинного ключа id (BIGINT, PK), зовнішніх ключів на товар product_id (BIGINT, FK) та користувача user_id (BIGINT, FK), числової оцінки rating (INTEGER) та текстового коментаря comment (TEXT).

Цей модуль графової аналітики складу є унікальною особливістю платформи і відповідає за збереження даних для математичної моделі графа інгредієнтів. Оскільки один товар може містити багато інгредієнтів, а один інгредієнт – використовуватися в багатьох товарах, тут реалізовано зв'язки типу "багато-до-багатьох" (Many-to-Many). Довідник усіх існуючих хімічних компонентів косметики представлено таблицею ingredients, яка має атрибути: id (BIGINT, PK), загальновживану назву name (VARCHAR), та детальний опис description (TEXT). Для збереження складу конкретного товару розроблено проміжну таблицю-зв'язку product_ingredients, що містить зовнішні ключі product_id (BIGINT, FK) та

ingredient_id (BIGINT, FK). Формування ребер графа, що позначають зв'язки між інгредієнтами на предмет їх сумісності, забезпечується таблицею ingredient_interactions. Вона включає первинний ключ id (BIGINT, PK), ідентифікатори пов'язаних компонентів ingredient_1_id та ingredient_2_id (BIGINT, FK), тип взаємодії interaction_type (VARCHAR, що приймає значення "synergy", "conflict", "neutral"), а також текстове пояснення причини такого впливу description (TEXT).

Модуль спільного кошика реалізує логіку електронної комерції та багатокористувацьких сесій покупок, фіксуючи їх стан у базі даних для забезпечення асинхронної роботи різних клієнтів. Таблиця ідентифікації сесій кошика (як індивідуальних, так і спільних) має назву carts і містить поля: id (BIGINT, PK), унікальний ідентифікатор uuid (UUID) для генерації посилання-запрошення, мітку часу створення created_at (TIMESTAMP) та поточний статус status (VARCHAR). Безпосередній перелік доданих товарів зберігається у проміжній таблиці cart_items, атрибутами якої є: первинний ключ id (BIGINT, PK), зовнішні ключі кошика cart_id та товару product_id (BIGINT, FK), кількість quantity (INTEGER) та ідентифікатор користувача, який додав цей товар у спільній сесії added_by_user_id (BIGINT, FK). Система купонів реалізована через таблицю promo_codes, що включає id (BIGINT, PK), унікальний код купона code (VARCHAR, UNIQUE), розмір знижки discount_percent (DECIMAL) та статус активності is_active (BOOLEAN). Фінальне збереження оформлених замовлень виконується у таблицю orders, яка має поля id (BIGINT, PK), зовнішні ключі user_id, cart_id та promo_code_id (BIGINT, FK), загальну суму до сплати total_amount (DECIMAL) та статус замовлення status (VARCHAR). Для збереження ціни товару на момент покупки (незалежно від майбутніх змін у каталозі) передбачено таблицю історичної фіксації позицій order_items, яка містить id (BIGINT, PK), зовнішні ключі order_id та product_id (BIGINT, FK), зафіксовану ціну price_at_purchase (DECIMAL) та кількість quantity (INTEGER).

2.3.3 Визначення зв'язків між таблицями та забезпечення цілісності

Важливим етапом проєктування бази даних стало встановлення реляційних зв'язків між визначеними таблицями за допомогою зовнішніх ключів (Foreign Keys). У базі даних реалізовано наступні типи зв'язків:

1. Один-до-багатьох (1:M): Найпоширеніший тип зв'язку в системі. Наприклад, одна категорія (categories) може містити багато товарів (products), але один товар належить лише до однієї категорії. Аналогічний зв'язок налаштовано між користувачами (users) та їхніми замовленнями (orders), а також між замовленнями та позиціями замовлень (order_items).

2. Багато-до-багатьох (M:N): Даний тип зв'язку не може бути безпосередньо реалізований у реляційній моделі, тому він логічно розбивається через створення проміжних таблиць. У системі «BeautyHeaven» цей підхід застосовано для реалізації складу товарів. Таблиці products та ingredients пов'язані через проміжну таблицю product_ingredients. Завдяки цьому один товар може містити багато інгредієнтів, а один інгредієнт - входити до складу багатьох товарів.

Для забезпечення посилальної цілісності даних на рівні бази даних було застосовано механізм каскадних обмежень (Constraints). Такий підхід дозволяє делегувати контроль узгодженості даних безпосередньо рушію СКБД, що мінімізує ризик програмних помилок на стороні бекенду [21].

Наприклад, при видаленні певного товару з таблиці products, налаштування ON DELETE CASCADE для зовнішнього ключа в таблиці product_ingredients автоматично видалить усі пов'язані записи про його склад, запобігаючи виникненню «осиротілих» даних у системі.

Результатом виконаних робіт стала оптимізована, нормалізована структура реляційної бази даних, яка повністю відповідає технічному завданню та готова до інтеграції з серверною частиною веб-додатка (через ORM-фреймворк). Детальний опис структури представлено у вигляді сутнісно-зв'язкової моделі.

ER-діаграма спроєктованої бази даних зображена в Додатку Б.

2.4 Програмування сайту

Завдання етапу програмування полягає у фізичній реалізації спроєктованої архітектури бази даних, створенні програмних інтерфейсів (API) та розробці інтерактивної клієнтської частини. Для забезпечення високої продуктивності, масштабованості та зручності підтримки коду, процес розробки було розділено на створення серверної (Backend) та клієнтської (Frontend) частин, які взаємодіють між собою за допомогою HTTP-запитів та протоколу WebSockets.

2.4.1 Написання клієнтської частини

Програмування клієнтської частини (Frontend) здійснювалося на базі бібліотеки React. Для ініціалізації проєкту, збирання модулів (Bundling) та забезпечення швидкого гарячого перезавантаження (Hot Module Replacement) під час розробки було використано сучасний інструмент збирання Vite. Завдяки використанню нативних ES-модулів у браузері, цей інструмент забезпечує значно вищу швидкість розробки порівняно з традиційними бандлерами [22]. Керування залежностями та встановлення зовнішніх пакетів здійснювалося за допомогою пакетного менеджера npm (Node Package Manager).

Основний акцент під час програмування клієнтської частини був зроблений на реалізації інтерактивних модулів, які забезпечують унікальний користувацький досвід: візуалізації графів та підтримці спільних сесій.

Для реалізації функціоналу Multi-user Cart архітектуру клієнтського додатка було розширено за допомогою патерну React Context. Візуальна частина реалізована у компоненті SharedCartPage.jsx, тоді як уся складна бізнес-логіка та мережева взаємодія інкапсульована у провайдері SharedCartContext.jsx. Динаміку взаємодії між користувачами, клієнтським додатком React, брокером повідомлень STOMP та сервером Spring Boot під час ініціалізації та синхронізації спільного кошика детально спроєктовано за допомогою UML-моделювання. Діаграма послідовності (Sequence) для цього процесу наведена в Додатку В.

Головною задачею цього контексту є встановлення двостороннього з'єднання із сервером (через бібліотеки SockJS та @stomp/stompjs) для отримання оновлень у режимі реального часу. При успішному підключенні (onConnect) клієнт підписується на унікальний канал кімнати (/topic/cart/{shareToken}). При надходженні нового повідомлення від сервера (наприклад, інший користувач додав товар), виконується парсинг JSON та оновлення стану кошика, що викликає миттєвий перерендер інтерфейсу у всіх учасників сесії, як показано в лістингу 2.1.

```
useEffect(() => {
   const socket = new SockJS('http://localhost:8080/ws');
   const stompClient = Stomp.over(socket);

   stompClient.connect({}, () => {
       stompClient.subscribe(`/topic/cart/${token}`, (message) => {
           const data = JSON.parse(message.body);
           setCartItems(data.items);
       }); }
```

Лістинг 2.1 – Фрагмент коду ініціалізації WebSocket-з'єднання у компоненті SharedCartPage

Для відображення зв'язків між хімічними компонентами косметики розроблено компонент CompabilityGraph.jsx, який використовує спеціалізовані бібліотеки (react-force-graph) для рендерингу графа.

Основна програмна логіка компонента полягає у гнучкому налаштуванні фізичного рушія та параметрів відмальовування на полотні (Canvas). За допомогою спеціалізованих функцій зворотного виклику (callbacks), таких як nodeCanvasObject та linkColor, компонент динамічно розраховує радіус бульбашок, їхній колір залежно від типу джерела, а також товщину та стиль ліній (ребер), що символізують синергію або конфлікт інгредієнтів, відповідно до лістингу 2.2.

```
<div className="relative overflow-hidden rounded-2xl border border-
zinc-100 bg-zinc-50">
        <ForceGraph2D
          ref={fgRef}
          graphData={graphData}
          width={390}
          height={280}
          nodeLabel={(n) => `${n.name} (${n.source})`}
                     linkColor={(l)   =>   LINK_COLORS[l.type]   ||
LINK_COLORS.NEUTRAL}
          linkWidth={(l) => l.isCrossProduct ? 2.5 : 1.2}
          linkLineDash={(l) => l.type === 'NEUTRAL' ? [4, 4] : null}
           onLinkHover={(l) => setTooltip(l ? { type: l.type, desc:
l.description, cross: l.isCrossProduct } : null)}
          nodeCanvasObject={(node, ctx, scale) => {
            const r    = Math.max(14 / scale, 8)
            const bg   = SOURCE_COLORS[node.source] || '#94a3b8'
            const fs   = Math.max(10 / scale, 7)

            ctx.arc(node.x, node.y, r, 0, 2 * Math.PI) }}
```

Лістинг 2.2 – Конфігурація рендерингу графа інгредієнтів на елементі Canvas

2.4.2 Написання серверної (Admin) частини та бізнес-логіки

Програмування серверної бізнес-логіки та API (Backend) виконувалося мовою Java з використанням фреймворку Spring Boot. Для ініціалізації проєкту застосовано сервіс Spring Initializr, а збирання проєкту та управління залежностями налаштовано за допомогою системи Maven. Архітектура серверної частини побудована за класичним шаблоном MVC (Model-View-Controller) з виділенням шарів доступу до даних (Repository), бізнес-логіки (Service) та маршрутизації запитів (Controller), що повністю відповідає кращим практикам розробки на базі екосистеми Spring [23].

Для забезпечення роботи адміністративної панелі (управління товарами та складом косметики) реалізовано набір захищених REST-контролерів. Наприклад, клас ProductController.java відповідає за обробку HTTP-запитів типу POST, PUT та DELETE. З метою забезпечення безпеки, контролю над серіалізацією JSON та відокремлення внутрішньої моделі бази даних від зовнішнього клієнта, у проєкті активно застосовується архітектурний патерн DTO (Data Transfer Object) [24]. Безпосередня обробка отриманих DTO-об'єктів, валідація вхідних даних та виконання основних бізнес-правил інкапсульовані у сервісному шарі (Service Layer) додатка. Своєю чергою, взаємодія цього шару з реляційною базою даних реалізована через абстракції Spring Data JPA, що забезпечує надійне та безпечне управління транзакціями. Фрагмент коду даного контролера наведено у лістингу 2.3.

```
@GetMapping("/{id}")
@Operation(summary = "Отримати товар за ID")
public ProductResponseDto getProductById(@PathVariable Long id) {
    return productService.getProductById(id);
}

@DeleteMapping("/{id}")
@Operation(summary = "Видалити товар")
public void deleteProduct(@PathVariable Long id) {
    productService.deleteProduct(id); }
```

Лістинг 2.3 – Фрагмент класу ProductController для обробки запитів на створення та видалення товару

Безпосередня бізнес-логіка валідації вхідних даних та взаємодії з базою даних інкапсульована у класах сервісного шару. Так, при збереженні нового товару разом з його складом (інгредієнтами), клас ProductService.java ініціює транзакцію до БД PostgreSQL за допомогою абстракцій Spring Data JPA, реалізацію яких наведено в лістингу 2.4.

```
@Transactional
public ProductResponseDto createProduct(ProductRequestDto dto) {
    Product product = new Product();
    product.setName(dto.getName());
    product.setDescription(dto.getDescription());
    product.setPrice(dto.getPrice());
    product.setImageUrl(dto.getImageUrl());
    product.setStockQuantity(dto.getStockQuantity());

    if (dto.getCategoryId() != null) {
        Category category =
categoryRepository.findById(dto.getCategoryId())
                .orElseThrow(() -> new
EntityNotFoundException("Category not found with id: " +
dto.getCategoryId()));
        product.setCategory(category);
    }
}
```

Лістинг 2.4 – Реалізація бізнес-логіки збереження товару у класі ProductService

Найскладнішим архітектурним завданням на серверній стороні стала реалізація багатокористувацької сесії кошика. Оскільки стандартний HTTP-протокол не підтримує ініціацію повідомлень від сервера до клієнта, було налаштовано підтримку WebSocket у Spring Boot. Для управління кімнатами кошиків розроблено клас CartWebSocketHandler, який використовує брокер повідомлень (STOMP) для ефективної маршрутизації подій у режимі реального часу [25]. Даний клас перехоплює повідомлення від одного клієнта (наприклад, подію «Товар додано»), оновлює стан кошика в базі даних через CartService, а потім здійснює розсилку (Broadcast) оновленого стану всім клієнтам, підключеним до даної сесії (кімнати), що продемонстровано в лістингу 2.5.

```
  @MessageMapping("/cart/{token}/update")
   public void cartUpdated(
           @DestinationVariable String token,
           @Payload CartEventMessage message
   ) {
            Set<String> users = roomUsers.getOrDefault(token, new
LinkedHashSet<>());
       CartEventMessage response = new CartEventMessage();
       response.setType(EventType.CART_UPDATED);
       response.setUserName(message.getUserName());
       response.setAction(message.getAction());
       response.setProductName(message.getProductName());
       response.setItems(message.getItems());
       response.setActiveUsers(snapshot(users));
           messagingTemplate.convertAndSend("/topic/cart/" + token,
response);
   }
}
```

Лістинг 2.5 – Серверна обробка WebSocket-повідомлень для синхронізації спільного кошика

Таким чином, у процесі програмування було успішно реалізовано складну клієнт-серверну архітектуру. Використання сучасних фреймворків (React, Spring Boot) та підходів (SPA, DTO, WebSockets) дозволило створити надійний, швидкий та зручний веб-додаток, який повністю реалізує закладений у технічному завданні функціонал.

2.5 Тестування web-сайту

2.5.1 Методологія та стратегія тестування

Тестування є невід'ємним етапом життєвого циклу розробки програмного

забезпечення, який гарантує відповідність розробленого продукту вимогам технічного завдання, виявляє дефекти на ранніх стадіях та забезпечує високий рівень користувацького досвіду (UX).

Зважаючи на клієнт-серверну архітектуру веб-додатка «BeautyHeaven», стратегія забезпечення якості (Quality Assurance) базувалася на комбінованому підході. Для перевірки працездатності системи застосовувалися такі методи:

1. Інтеграційне тестування API (Backend): перевірка серверної бізнес-логіки ізольовано від клієнтського інтерфейсу.
2. Ручне функціональне тестування (Frontend): перевірка позитивних та негативних сценаріїв взаємодії користувача з графічним інтерфейсом.
3. Кросбраузерне тестування: перевірка коректності відображення та роботи сайту в різних веб-браузерах.

2.5.2 Інтеграційне тестування серверної частини (API) за допомогою Postman

Оскільки клієнтська частина взаємодіє з сервером виключно через REST API, критично важливим було протестувати серверні кінцеві точки (Endpoints) незалежно від графічного інтерфейсу. Для цього використовувалося спеціалізоване середовище розробки та тестування API - Postman.

Окрім перевірки позитивних сценаріїв, значна увага приділялася негативному тестуванню на рівні API та автоматизації перевірок. Зокрема, у середовищі Postman було використано функціонал вбудованих скриптів (Postman Tests) для автоматичної валідації структури JSON-відповідей та контролю часу обробки запитів (Response time).

У процесі тестування було створено колекцію запитів, яка імітує повний життєвий цикл роботи користувача у системі. Були успішно виконані та перевірені наступні ключові сценарії:

1. Тестування управління контентом: Використовуючи отриманий JWT-токен, було виконано POST-запит від імені адміністратора на додавання нового

косметичного товару. Сервер повернув статус 201 Created, що підтверджує коректну роботу модуля збереження товарів та інгредієнтів.

2. Тестування бізнес-логіки покупок: Виконано комплексний запит на успішне створення замовлення, який підтвердив, що сервер коректно розраховує підсумкову суму, змінює статус кошика та фіксує позиції в історичній таблиці order_items.

3. Тестування безпеки та авторизації: Виконано POST-запит на маршрут реєстрації нового користувача. Перевірено збереження даних у БД. Після цього виконано запит на авторизацію (Login). Сервер успішно валідував дані та повернув у тілі відповіді згенерований JWT-токен (див. рис. 2.5).

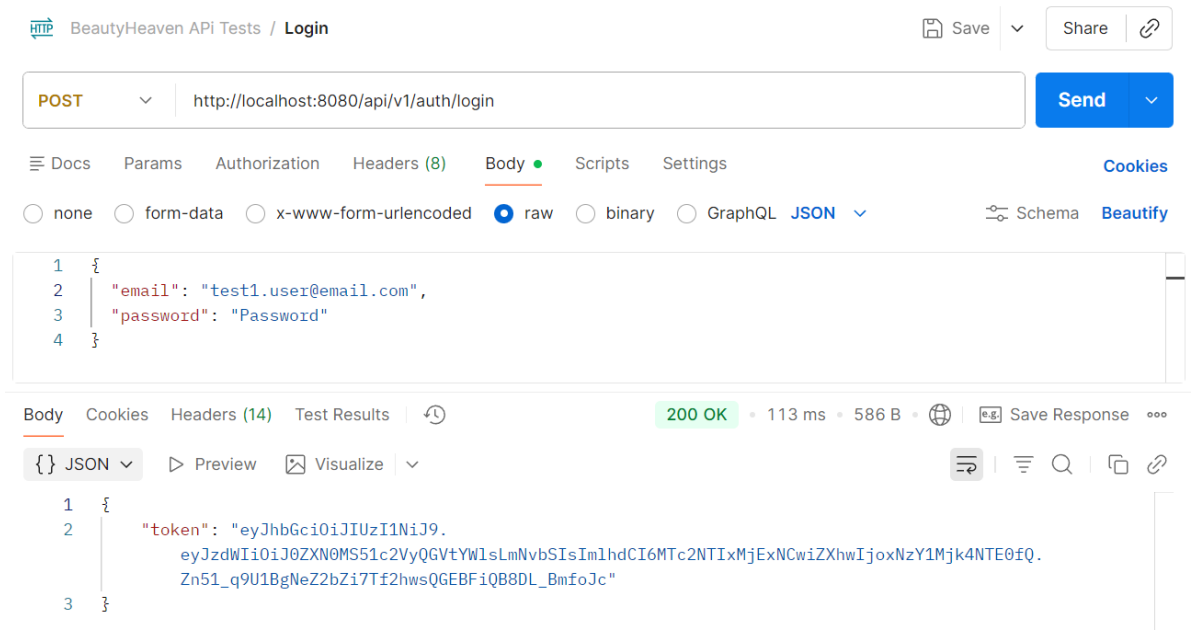


Рисунок 2.5 – Проведення інтеграційного тестування REST API за допомогою Postman

Отримані під час автоматизованого тестування результати, зафіксовані на ілюстрації, підтвердили повну відповідність серверної архітектури Spring Boot розробленим специфікаціям API. Усі надіслані запити повернули очікувані структури даних та HTTP-статуси відповіді у межах встановлених часових лімітів, що доводить надійність бізнес-логіки бекенду та його повну готовність до інтеграції з клієнтським інтерфейсом додатка.

2.5.3 Функціональне та кросбраузерне тестування клієнтського інтерфейсу

Кросбраузерне тестування проводилося з метою перевірки сумісності згенерованих JSX-шаблонів, CSS-стилів (Tailwind) та JavaScript-логіки у різних середовищах. Перевірка здійснювалася у трьох найпопулярніших сучасних браузерах: Google Chrome, Mozilla Firefox та Microsoft Edge. Під час тестування не було виявлено критичних візуальних дефектів чи розбіжностей у рендерингу графа інгредієнтів.

Функціональне тестування проводилося методом "чорного ящика" (Black-box testing) і включало перевірку як позитивних, так і негативних сценаріїв.

Позитивні сценарії підтвердили, що користувач може безперешкодно пройти шлях від перегляду каталогу до успішної оплати замовлення. Особлива увага приділялася тестуванню модуля "Спільний кошик" (WebSockets). Для цього було зімітовано паралельну роботу: сесія спільного кошика була відкрита одночасно у браузерах Chrome та Edge під різними обліковими записами. Тестування підтвердило, що додавання або видалення товару в одному браузері миттєво (з затримкою < 1 сек) відображається в іншому браузері без перезавантаження сторінки.

Негативні сценарії були спрямовані на перевірку системи обробки виняткових ситуацій (Error Handling) та валідації введених даних. Було перевірено, що система не «падає» при некоректних діях користувача, а видає зрозумілі текстові підказки. Зокрема, успішно протестовано виведення наступних повідомлень про помилки:

1. При спробі входу з незареєстрованим email або неправильним паролем: «Неправильний логін або пароль».
2. При спробі переходу до оформлення замовлення без доданих товарів: «Кошик порожній».
3. При спробі застосування недійсного або протермінованого купона у

спільному кошику: «Промокод недійсний або неактивний» (див. рис. 2.6).

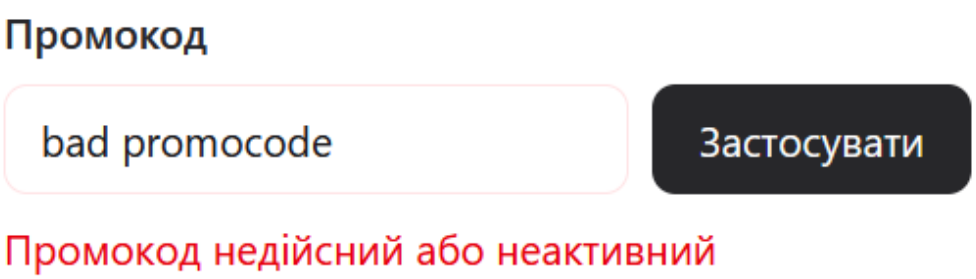


Рисунок 2.6 - Обробка системою негативних сценаріїв (виведення повідомлення про помилку)

Таким чином, проведене візуальне та функціональне моделювання інтерфейсу підтвердило гнучкість розробленої архітектури користувацького шару. Належне функціонування механізмів перехоплення винятків та валідації вхідних форм, що проілюстровано на наведеному макеті екранної форми, дозволяє мінімізувати когнітивне навантаження на кінцевого користувача під час виникнення позаштатних ситуацій та забезпечує високу стійкість клієнтської частини додатка React.

2.5.4 Контрольний список (Checklist) тестування компонентів веб-сайту

Для більш повного відображення критеріїв тестування та фіксації результатів ручної перевірки модулів, було розроблено формалізований контрольний список (Checklist) тестування системи. Даний інструмент дозволив гарантувати повне покриття функціоналу, описаного у технічному завданні. Першим етапом стала перевірка базових підсистем доступу користувачів, навігації та роботи інтерактивного каталогу косметичних засобів, результати якої наведено в таблиці 2.1.

Таблиця 2.1 – Контрольний список тестування підсистем авторизації та каталогу

Модуль / Підсистема	**Дія (Опис Тест-кейсу)**	**Очікуваний результат**	**Статус**
1. Авторизація та Безпека	Введення коректних даних у форму логіну.	Успішний вхід, перенаправлення на головну сторінку, токен збережено.	Успішно
	Введення невірного пароля.	Відмова у доступі, поява помилки «Неправильний логін або пароль».	Успішно
	Натискання кнопки «Вийти» (Logout).	Сесія знищується, користувач перенаправляється на сторінку логіну.	Успішно
2. Каталог та Товари	Застосування фільтра за брендом/категорією.	Каталог миттєво оновлюється, відображаючи лише релевантні товари.	Успішно
	Натискання на товар у каталозі.	Відкривається сторінка деталей товару з ціною.	Успішно

Як свідчать результати, наведені у таблиці 2.1, базовий функціонал доступу та візуалізації даних працює безперебійно. Після підтвердження стабільності цих модулів, наступним кроком стало тестування транзакційної логіки, а саме підсистем управління індивідуальними та спільними покупками, що відображено в таблиці 2.2.

Таблиця 2.2 – Контрольний список тестування функціоналу кошиків та замовлень

Модуль / Підсистема	**Дія (Опис Тест-кейсу)**	**Очікуваний результат**	**Статус**
1. Звичайний кошик	Додавання товару з каталогу в кошик.	Збільшується лічильник товарів у Header, товар з'являється у кошику.	Успішно
	Зміна кількості товару в кошику (+/-).	Загальна сума кошика перераховується автоматично.	Успішно
2. Спільний кошик	Генерація посилання на спільний кошик.	Система створює унікальний URL та копіює його в буфер обміну.	Успішно
	Перехід за посиланням іншого користувача.	Користувач приєднується до кімнати, його аватар/ім'я з'являється у панелі «Live».	Успішно

Результати тестування, подані у таблиці 2.2, підтверджують високу надійність роботи двосторонніх з'єднань (WebSockets) та правильність розрахунку фінансових показників у режимі реального часу. Завершальним етапом ручного контролю якості стала перевірка закритої частини системи (адміністративної панелі) та її загальної сумісності з браузерами, результати якої зведено в таблицю 2.3.

Таблиця 2.3 – Контрольний список тестування адміністративної панелі та кросбраузерності

Модуль / Підсистема	**Дія (Опис Тест-кейсу)**	**Очікуваний результат**	**Статус**
1.Адміністративна панель	Додавання нового товару через форму (POST).	Товар зберігається в БД та одразу стає доступним у публічному каталозі.	Успішно
	Видалення існуючого товару (DELETE).	Спрацьовує каскадне видалення, товар зникає з системи.	Успішно
2.Кросбраузерність	Перевірка верстки (CSS) у Firefox та Edge.	Елементи не накладаються один на одного, всі кнопки клікабельні.	Успішно

Дані таблиці 2.3 підтверджують, що система коректно обробляє CRUD-операції та зберігає візуальну цілісність у різних середовищах виконання. Загалом, за результатами проведеного багатоступеневого тестування (API, кросбраузерність, ручне функціональне тестування) можна зробити висновок, що програмний продукт повністю відповідає заявленим вимогам. Система демонструє стабільну роботу, коректно обробляє помилкові дії користувачів та забезпечує надійну синхронізацію даних. Розроблений вебсайт інтернет-магазину косметики «BeautyHeaven» успішно пройшов усі етапи перевірки та готовий до експлуатації.

Усі виявлені під час проходження контрольного списку незначні дефекти були оперативно усунуті на етапі налагодження. Відсутність критичних вразливостей та висока швидкість відгуку клієнтського інтерфейсу дозволяють безпечно перейти до фінальної стадії – розгортання додатка на робочих хмарних серверах для надання доступу реальним користувачам.

3. СПЕЦІАЛЬНИЙ РОЗДІЛ

3.1 Розміщення сайту в Інтернеті

Для забезпечення цілодобового доступу користувачів до розробленого веб-додатку в глобальній мережі Інтернет було обрано сучасну розподілену хмарну архітектуру розгортання. Оскільки система побудована за принципом розділення клієнтської (Frontend) та серверної (Backend) частин, процес публікації вимагав незалежного розгортання трьох ключових компонентів: бази даних, серверного API та користувацького інтерфейсу.

Такий підхід забезпечує високу відмовостійкість, гнучке масштабування окремих вузлів та дозволяє використовувати спеціалізовані PaaS-платформи (Platform as a Service) для кожного стеку технологій. Фізичний розподіл компонентів системи по хмарних серверах, архітектуру мережевих взаємозв'язків та протоколи передачі даних було візуалізовано на етапі системного проєктування.

3.1.1 Хмарне розміщення бази даних

Для зберігання даних було обрано хмарну платформу Neon.tech, яка надає послуги Serverless PostgreSQL. Цей вибір обґрунтований автоматичним масштабуванням обчислювальних ресурсів та високим рівнем безпеки.

Процедура розгортання бази даних включала наступні етапи:

1. Реєстрація на платформі та створення нового проєкту з вибором регіону сервера (Frankfurt, EU) для мінімізації мережевої затримки для цільової аудиторії.
2. Отримання автентифікаційних даних: унікального домену сервера (Host), імені користувача та згенерованого пароля.
3. Підключення до створеної хмарної бази через локальний клієнт pgAdmin для імпорту початкової структури таблиць та тестових даних. Для забезпечення безпеки підключення здійснювалося з обов'язковим використанням зашифрованого з'єднання (sslmode=require).

Інтерфейс отримання параметрів підключення хмарної бази даних наведено на рисунку 3.1.

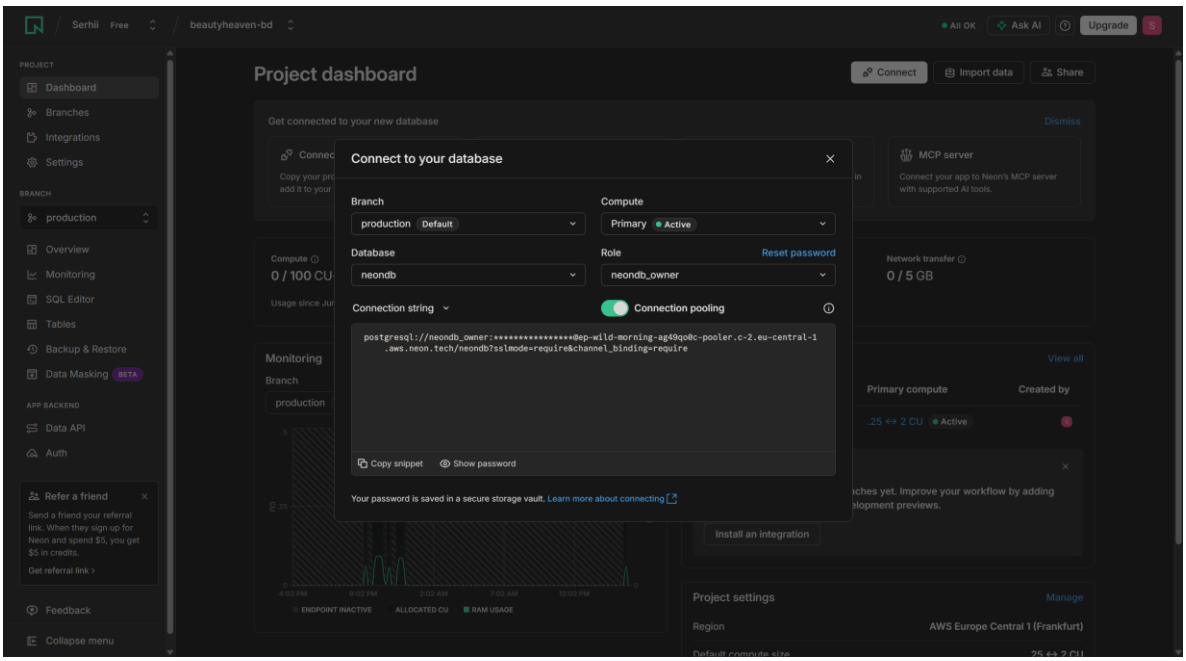


Рисунок 3.1 – Налаштування підключення до хмарної бази даних Neon

Отримані автентифікаційні дані дозволили успішно підключитися до хмарної бази через локальний клієнт pgAdmin для експорту початкової структури таблиць та тестових даних. Для забезпечення максимального рівня безпеки налаштування підключення здійснювалося з обов'язковим використанням зашифрованого з'єднання (параметр sslmode=require). Таким чином, базу даних було повністю підготовлено для подальшої інтеграції з серверною частиною системи.

3.1.2 Розгортання серверної частини (Backend)

Серверна частина, розроблена на базі фреймворку Java Spring Boot, була розміщена на хмарній платформі Render.com. Для забезпечення передбачуваності середовища виконання та уникнення конфліктів версій Java, розгортання проводилося за допомогою технології контейнеризації Docker.

Процедура розміщення бекенду:

1. Контейнеризація: У кореневому каталозі проєкту було створено файл Dockerfile, який реалізує багатоетапну збірку (Multi-stage build). На першому етапі інструмент Maven компілює вихідний код у виконуваний .jar файл, а на другому — запускає його в середовищі JRE.

2. Налаштування CI/CD: Платформу Render було синхронізовано з репозиторієм проєкту на GitHub. Це дозволило налаштувати безперервну інтеграцію: при кожному оновленні коду в головній гілці автоматично ініціюється процес нової збірки.

3. Конфігурація змінних оточення: Паролі та ключі доступу не зберігалися у вихідному коді. У панелі керування Render було створено системні змінні: DB_URL, DB_USER та DB_PASS.

Після успішного завершення збірки статус сервісу змінюється на активний, а в консолі фіксується успішний старт програми, що підтверджує коректну роботу серверної частини (див. рис. 3.2).

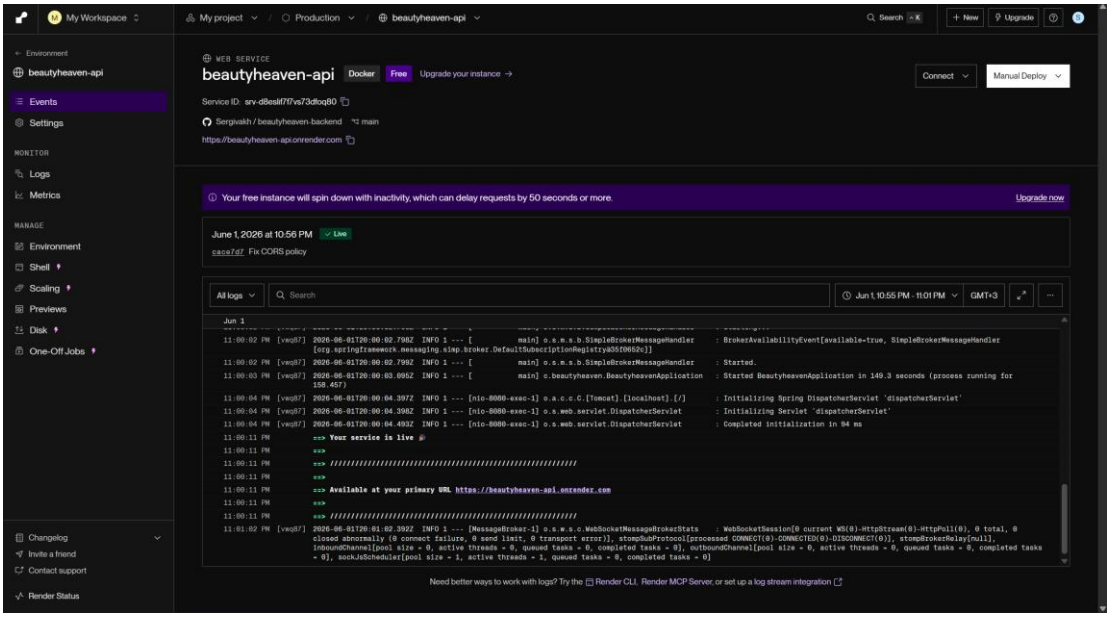


Рисунок 3.2 – Успішне розгортання серверної частини API на платформі Render

Наявність активного статусу та відсутність помилок у логах свідчать про те, що Docker-контейнер з додатком успішно зібрано, а підключення до хмарної бази даних Neon встановлено коректно.

3.1.3 Розгортання клієнтської частини (Frontend)

Для хостингу користувацького інтерфейсу, розробленого за допомогою бібліотеки React та збирача Vite, було використано платформу Vercel.

Основні етапи публікації фронтенду:

1. Синхронізація з репозиторієм: Проєкт було імпортовано на платформу безпосередньо з GitHub для автоматизації розгортання.

2. Адаптація маршрутизації API: У конфігурації HTTP-клієнта (Axios) було налаштовано динамічне визначення базової адреси сервера, що направляє запити на публічну адресу API Render-сервера.

3. Налаштування серверної маршрутизації SPA: Для коректної роботи прямих посилань на внутрішні сторінки у корінь проєкту було додано конфігураційний файл vercel.json, який інструктує сервер перенаправляти всі вхідні запити на головний файл index.html.

Успішне розгортання клієнтської частини та автоматичне призначення доменного імені на платформі Vercel продемонстровано на рисунку 3.3.

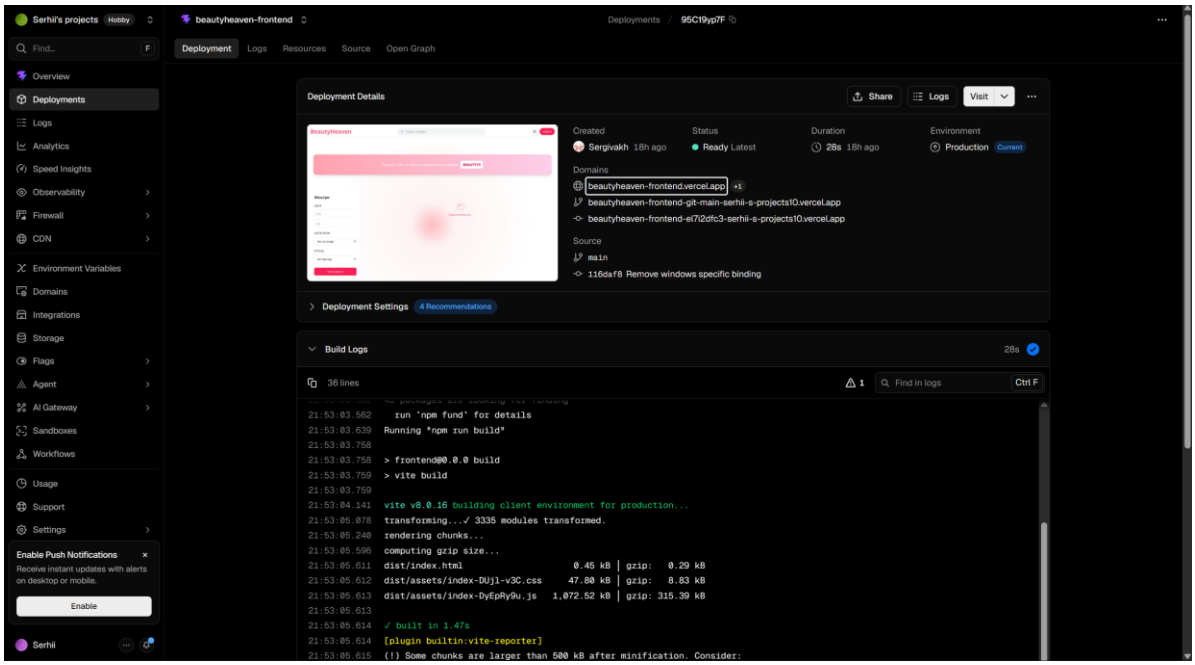


Рисунок 3.3 – Завершення розгортання клієнтської частини на платформі Vercel

Присвоєний платформою Vercel домен став офіційною публічною адресою розробленого веб-додатку. Завдяки автоматичній генерації SSL-сертифікатів, платформа також забезпечила захищене HTTPS-з'єднання, що є обов'язковою вимогою для сучасних комерційних веб-ресурсів та гарантує безпеку обміну даними.

3.1.4 Перевірка роботи сайту за доменним ім'ям

Після завершення процесів компіляції та розгортання, платформа Vercel автоматично згенерувала та безкоштовно надала проекту доменне ім'я https://beautyheaven-frontend.vercel.app/, забезпечене автоматичним SSL-сертифікатом для підтримки безпечного протоколу HTTPS.

Завершальним етапом стала комплексна перевірка працездатності розгорнутої системи в реальних умовах. Результат роботи розгорнутого веб-додатку за публічною адресою наведено на рисунку 3.4.

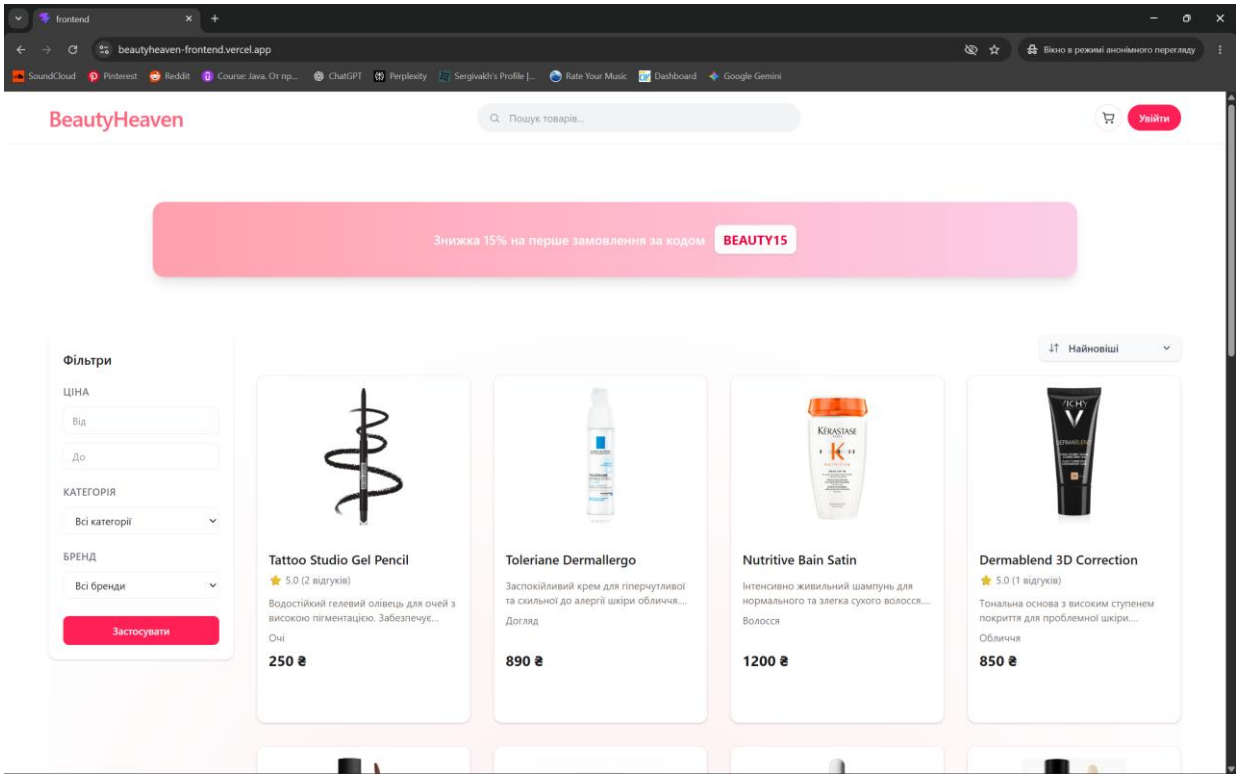


Рисунок 3.4 – Головна сторінка веб-додатку, доступного в мережі Інтернет

Перевірка підтвердила доступність інтерфейсу та успішну інтеграцію компонентів: клієнтський додаток коректно надсилає GET-запити на домен бекенду, який звертається до бази даних PostgreSQL та повертає необхідні дані для відображення. Тестування функціоналу кошика та авторизації довело, що обрана архітектура забезпечує надійну та швидку роботу сервісу з можливістю подальшого масштабування.

3.2 Інструкція з обслуговування та наповнення сайту

Для забезпечення ефективного функціонування розробленого веб-додатку в умовах реальної експлуатації необхідно підтримувати актуальність бази даних та контенту. Оскільки система побудована на базі сучасних веб-технологій, процес адміністрування та наповнення сайту здійснюється через зручний графічний інтерфейс користувача (адміністративну панель), що не вимагає від персоналу глибоких знань у галузі програмування чи написання SQL-запитів.

3.2.1 Мінімальна конфігурація сервера для хостингу сайту

Оскільки проєкт використовує мікросервісну архітектуру, поняття «сервер» розділяється на три незалежні хмарні компоненти. Для стабільної роботи системи визначено такі мінімальні технічні вимоги до апаратного та програмного забезпечення хостинг-провайдерів:

1. Вузол бази даних (СУБД PostgreSQL 15+ у хмарі Neon):
   - Процесор (CPU): архітектура x86_64, процесор серверного класу рівня Intel Xeon Platinum 8259CL або еквівалент AMD EPYC 7571. Базова тактова частота: від 2.5 ГГц.
   - Оперативна пам'ять (RAM): 1 ГБ (1024 МБ) типу DDR4 ECC (Error-Correcting Code) з тактовою частотою 2666 МГц. Наявність ECC є критичною вимогою для СУБД з метою запобігання пошкодженню

даних у пам'яті.

- Дискова підсистема (Storage): твердотільний накопичувач стандарту NVMe SSD (інтерфейс PCIe 3.0 x4) об’ємом від 5 ГБ. Мінімальна швидкість послідовного читання/запису – 1500 МБ/с.
- Мережевий інтерфейс: Ethernet-контролер із пропускною здатністю 1 Гбіт/с.

2. Вузол серверного додатку (Бекенд на Java Spring Boot / Render.com):
   - Процесор (CPU): архітектура x86_64 або ARM64, модель рівня AWS Graviton2 або Intel Xeon Scalable (наприклад, Intel Xeon E5-2686 v4), тактова частота від 2.3 ГГц.
   - Оперативна пам'ять (RAM): 512 МБ типу DDR4 ECC з частотою від 2400 МГц. Даний об'єм є мінімально необхідним для ініціалізації середовища Java Runtime Environment (JRE) та роботи ізольованого Docker-контейнера.
   - Дискова підсистема (Storage): Enterprise SSD (інтерфейс SATA III 6 Гбіт/с або SAS) об’ємом від 10 ГБ для зберігання образу ОС Linux (Ubuntu 22.04 LTS), логів програми та згенерованих файлів.
3. Вузол доставки клієнтського контенту (Edge Node фронтенду / Vercel):
   - Оскільки клієнтська частина після збірки (build) за допомогою Vite перетворюється на оптимізований набір статичних файлів (HTML, CSS, JS), її розгортання відбувається на магістральних серверах CDN (Content Delivery Network).
   - Оперативна пам'ять вузла: від 128 МБ високошвидкісної кеш-пам'яті на периферійному сервері для миттєвої віддачі статичних ресурсів.
   - Мережеві вимоги: підключення сервера до оптичного магістрального каналу зв'язку від 10 Гбіт/с, апаратна підтримка протоколів HTTP/2, HTTP/3, IPv6 та автоматичне шифрування трафіку за стандартом TLS 1.3.

3.2.2 Основні функції та меню адміністративної частини сайту

Для доступу до функцій керування контентом користувач повинен пройти процес авторизації, використовуючи обліковий запис із призначеною роллю адміністратора (role = ADMIN). Після успішного входу в систему інтерфейс веб-додатку адаптується: в головному навігаційному меню з'являється додатковий розділ «Адміністративна панель».

Основне меню адміністративної частини розділено на кілька логічних блоків:

1. Управління товарами: перегляд загального списку продукції, додавання, редагування або видалення позицій.
2. Управління категоріями: створення та зміна ієрархії розділів каталогу для зручної навігації.
3. Управління брендами: ведення довідника виробників косметики.
4. Аналітика та замовлення: перегляд історії покупок користувачів.
5. Управління інгредієнтами та зв’язками між ними: створення та зміна інгредієнтів і їх взаємодій.

Зовнішній вигляд головного екрана адміністративної панелі з переліком доступних функцій наведено на рисунку 3.5.

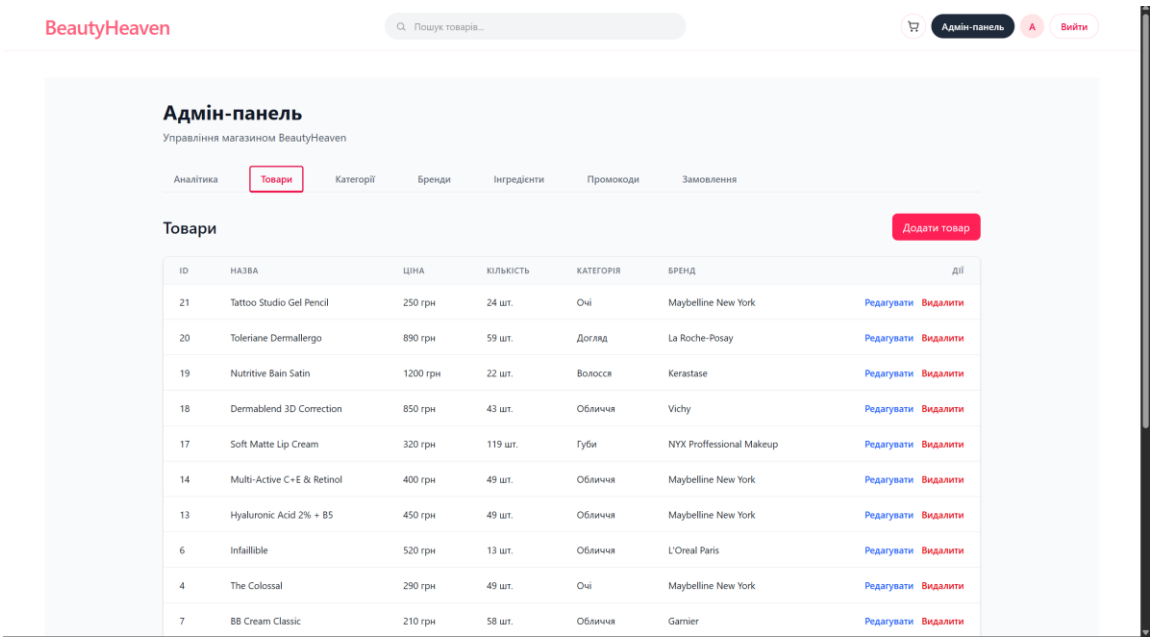


Рисунок 3.5 – Інтерфейс адміністративної частини веб-додатку

Така структура меню дозволяє адміністратору швидко орієнтуватися в системі. Завдяки реалізації концепції Single Page Application (SPA), перехід між розділами панелі керування відбувається миттєво, без перезавантаження сторінки, що значно пришвидшує роботу контент-менеджера.

3.2.3 Інструкція з редагування існуючого та створення нового контенту

Процес наповнення сайту має строгу послідовність, зумовлену реляційною структурою бази даних. Оскільки кожен товар повинен належати до певної категорії та бренду, адміністратор має спочатку заповнити відповідні довідники, і лише після цього переходити до створення карток товарів.

Створення брендів та категорій:

1. Необхідно перейти у розділ «Бренди» або «Категорії».
2. Натиснути кнопку «Додати бренд/категорію».
3. У модальному вікні заповнити текстове поле з назвою (наприклад, найменування виробника або тип косметики).
4. Зберегти зміни. Дані миттєво відправляються POST-запитом на сервер та зберігаються у базі даних.

Створення нового товару: для додавання нової позиції в каталог адміністратор повинен перейти у розділ «Товари» та натиснути кнопку «Додати товар». Відкриється електронна форма, яка містить усі необхідні поля для формування картки товару. У цій формі адміністратор послідовно заповнює базові характеристики косметичного засобу: вказує назву, актуальну ціну, додає текстовий опис та завантажує зображення. Також необхідно обрати зі спадних списків раніше створені категорію та бренд, і обов'язково призначити перелік інгредієнтів. Після заповнення всіх полів натискається кнопка збереження, і новий товар миттєво з'являється у клієнтському каталозі. (див. рис. 3.6).

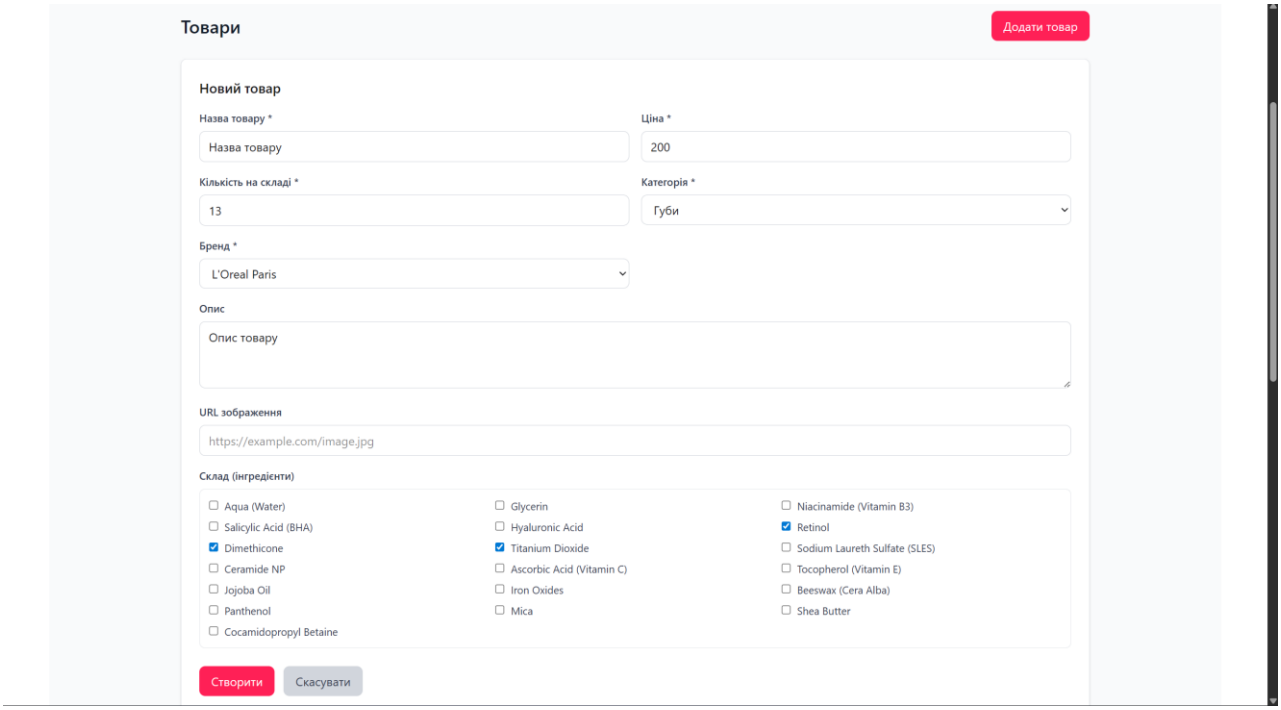


Рисунок 3.6 – Форма додавання нового товару в каталог

Алгоритм заповнення форми:

1. Основна інформація: введення назви товару та детального текстового опису характеристик.

2. Ціноутворення та облік: встановлення актуальної ціни та кількості одиниць товару на складі (Stock Quantity).

3. Класифікація: вибір відповідної категорії та бренду із випадаючих списків, дані для яких автоматично підвантажуються з бази даних.

4. Медіаконтент: вказати URL-посилання на зображення.

5. Після перевірки введених даних необхідно натиснути кнопку «Створити». Фронтенд-додаток валідує дані та відправляє їх на бекенд для фіксації в системі.

Редагування та видалення контенту: для підтримки актуальності інформації передбачено механізми редагування. У загальному списку товарів біля кожної позиції розміщено кнопки керування, як показано на рисунку 3.7.

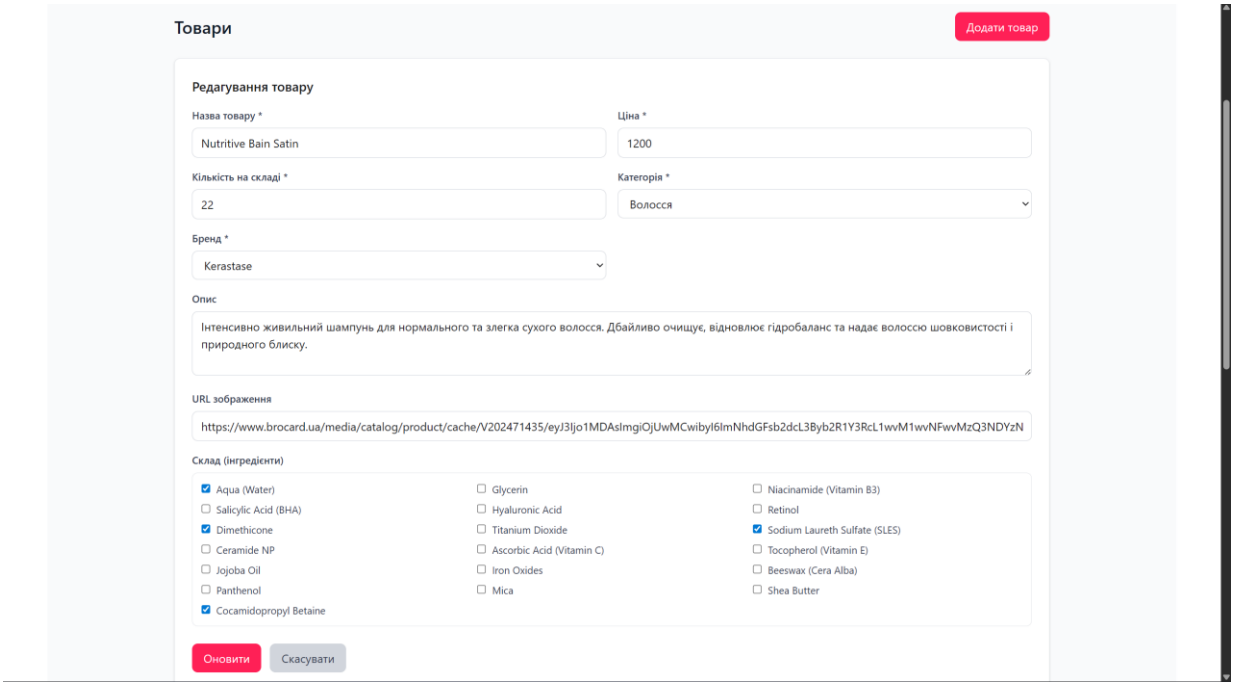


Рисунок 3.7 – Інструменти управління існуючим контентом

При натисканні на кнопку «Редагувати» відкривається форма, ідентична формі створення, але вже заповнена поточними даними товару. Адміністратор може змінити ціну, опис чи залишок на складі та зберегти оновлення (відбувається PUT-запит до API). У разі необхідності вилучення товару з асортименту використовується кнопка «Видалити». Перед фізичним видаленням запису з бази даних (DELETE-запит), система генерує діалогове вікно з проханням підтвердити дію, що запобігає випадковій втраті інформації.

Таким чином, розроблений інтерфейс адміністративної панелі забезпечує повний цикл управління контентом (CRUD — Create, Read, Update, Delete) у зручному та інтуїтивно зрозумілому форматі. Реалізовані механізми валідації форм та захисту від випадкових дій мінімізують ризик людської помилки під час обслуговування веб-додатку.

3.3 Інструкція з популяризації та підтримки сайту

Успішне розгортання веб-додатку в мережі Інтернет є лише першим етапом життєвого циклу комерційного програмного продукту. Для залучення цільової

аудиторії, конвертації відвідувачів у покупців та забезпечення стабільного функціонування платформи «Beauty Heaven» необхідно реалізувати комплексну стратегію цифрового маркетингу та безперервної технічної підтримки. Процес популяризації інтернет-магазину косметики вимагає використання синергії інструментів пошукової оптимізації, платного трафіку та соціальних комунікацій.

3.3.1 Пошукова оптимізація (SEO) та технічна адаптація

Пошукова оптимізація (Search Engine Optimization) є фундаментом для отримання безкоштовного трафіку з пошукових систем, таких як Google. Оскільки клієнтська частина розроблена на базі бібліотеки React (архітектура Single Page Application), контент генерується динамічно на стороні клієнта за допомогою JavaScript. Для забезпечення коректного індексування такого сайту пошуковими роботами необхідно впровадити ряд технічних рішень.

По-перше, здійснюється налаштування метатегів (Title, Description, Keywords) для кожної сторінки каталогу. Оскільки асортимент інтернет-магазину косметики є візуально орієнтованим, критично важливим є заповнення атрибутів alt для всіх зображень товарів, що дозволяє отримувати трафік із розділу Google Зображення. По-друге, впроваджується семантична мікророзмітка за стандартом Schema.org. Це дозволяє пошуковій системі формувати розширені сніпети (Rich Snippets), виводячи ціну, рейтинг та наявність товару безпосередньо у результатах пошуку (див. рис. 3.8).

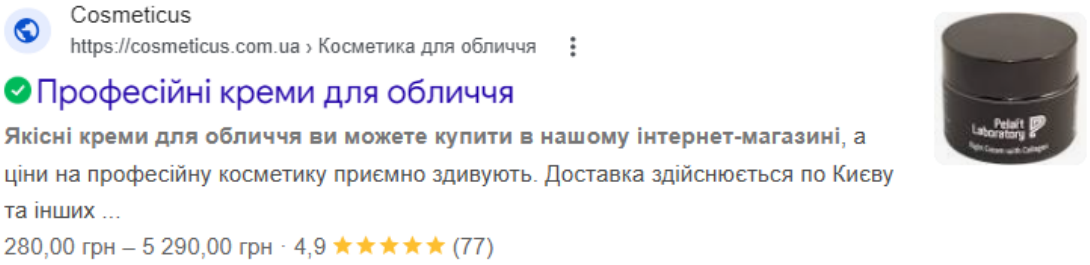


Рисунок 3.8 – Структура розширеного пошукового сніпету товару

Впровадження мікророзмітки та оптимізація метатегів суттєво підвищують показник клікабельності (CTR — Click-Through Rate) у пошуковій видачі. Крім технічної частини, оптимізація включає роботу з семантичним ядром: створення унікальних SEO-описів для категорій товарів (наприклад, «креми для обличчя», «декоративна косметика») із використанням релевантних ключових слів, що відповідають пошуковим запитам потенційних клієнтів.

3.3.2 Контекстна та таргетована реклама (PPC)

Для швидкого залучення аудиторії на етапі запуску проєкту використовується модель оплати за клік (Pay-Per-Click). Найбільш конверсійним інструментом для e-commerce є товарні оголошення.

Для налаштування таких рекламних кампаній генерується динамічний фід даних – структурований масив інформації про всі товари (назва, ціна, наявність, посилання на фото). Замість ручного формування файлів, серверна частина проєкту автоматично генерує необхідну структуру через REST API. Структура даних у форматі JSON, що генерується бекендом для формування динамічного товарного фіду, продемонстрована на рисунку 3.9.


```
{"content":[{"id":21,"name":"Tattoo Studio Gel Pencil","description":"Водостійкий гелевий олівець для очей з високою пігментацією. Забезпечує насичений колір, який тримається до 36 годин без розмазування та відбивання на повіках.","price":250.00,"imageUrl":"https://m.media-amazon.com/images/I/61AHifEcFVL.jpg","stockQuantity":24,"categoryId":2,"brandId":2,"categoryName":"Очі","brandName":"Maybelline New York","averageRating":5.0,"reviewCount":2},{"id":20,"name":"Toleriane Dermallergo","description":"Заспокійливий крем для гіперчутливої та схильної до алергії шкіри обличчя. Зменшує почервоніння, миттєво знімає подразнення та інтенсивно зволожує.","price":890.00,"imageUrl":"https://u.makeup.com.ua/r/rt/rtl5kolqsguq.jpg","stockQuantity":59,"categoryId":4,"brandId":5,"categoryName":"Догляд","brandName":"La Roche-Posay","averageRating":0.0,"reviewCount":0},{"id":19,"name":"Nutritive Bain Satin","description":"Інтенсивно живильний шампунь для нормального та злегка сухого волосся. Дбайливо очищує, відновлює гідробаланс та надає волоссю шовковистості і природного блиску.","price":1200.00,"imageUrl":"https://www.brocard.ua/media/catalog/product/cache/V202471435/eyJ3Ijo1MDAsImgiOjUwMCwibyI6ImNhdGFsb2dcL3Byb2R1Y3RcL1wvM1wvNFwvMzQ3NDYzNzE1NDkxMl8xLmpwZyJ9/kerastase-nutritive.webp","stockQuantity":22,"categoryId":5,"brandId":8,"categoryName":"Волосся","brandName":"Kerastase","averageRating":0.0,"reviewCount":0},{"id":18,"name":"Dermablend 3D Correction","description":"Тональна основа з високим ступенем покриття для проблемної шкіри. Вирівнює рельєф, маскує недоліки та забезпечує ідеально матовий фініш до 16 годин.","price":850.00,"imageUrl":"https://cdn.notinoimg.com/detail_main_mq/vichy/3337871316617-o/dermablend___130904.jpg","stockQuantity":43,"categoryId":1,"brandId":6,"categoryName":"Обличчя","brandName":"Vichy","averageRating":5.0,"reviewCount":1},{"id":17,"name":"Soft Matte Lip Cream","description":"Легендарна матова помада-крем з ніжною текстурою. Забезпечує стійке покриття, не пересушує губи та залишає приємний солодкий аромат на весь
```


Рисунок 3.9 – Структура даних (JSON) для формування динамічного товарного фіду

Автоматизація оновлення фіду через API гарантує, що рекламні платформи отримують виключно актуальні дані. Паралельно можуть запускатися таргетовані рекламні кампанії в соціальних мережах, зокрема механізми динамічного ретаргетингу, які показують персоналізовану рекламу користувачам, що не завершили процес оформлення замовлення.

3.3.3 Просування через соціальні мережі (SMM) та контент-маркетинг

Специфіка ніші товарів для краси вимагає сильної візуальної присутності в соціальних мережах, що відіграє ключову роль у формуванні довіри до бренду. Для інтеграції зовнішніх платформ із сайтом налаштовується протокол Open Graph (OG-теги).

Це гарантує, що при надсиланні посилання на товар у месенджерах (наприклад, Telegram або Viber) автоматично формується візуально привабливе прев'ю з фотографією, назвою та описом ресурсу. Приклад коректного відображення картки товару завдяки OG-тегам при поширенні посилання у месенджері наведено на рисунку 3.10.

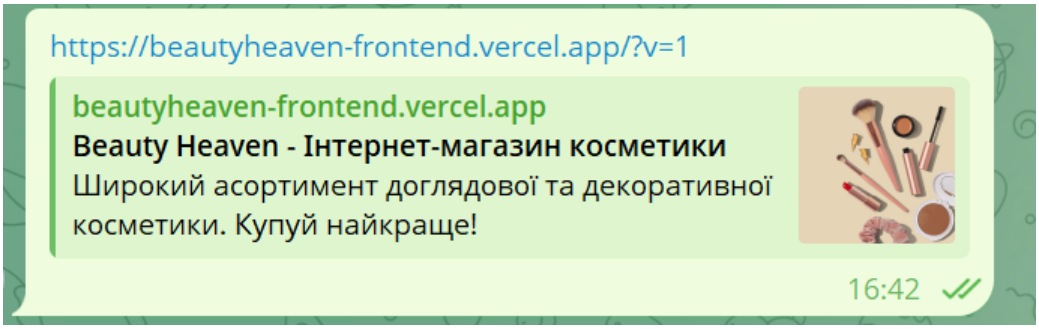


Рисунок 3.10 – Відображення картки товару при соціальному шерингу

Коректне налаштування Open Graph не лише покращує естетичне сприйняття посилань, але й підвищує рівень довіри користувачів до ресурсу, стимулюючи їх переходити на сайт із соціальних мереж та месенджерів.

3.3.4 Технічна підтримка та системний моніторинг

Популяризація неможлива без підтримки технічної стабільності сервера та аналізу показників роботи системи. З технічної точки зору, підтримка сайту включає постійний моніторинг мережевого навантаження та обсягу переданих даних. Цей показник є критично важливим, оскільки він прямо корелює з кількістю активних користувачів на сайті: зростання вихідного трафіку свідчить про збільшення кількості запитів до каталогу та підтверджує ефективність проведених маркетингових кампаній.

Платформа хмарного хостингу надає вбудовані аналітичні інструменти для відстеження стану віртуальної машини та обсягу згенерованого вихідного трафіку в режимі реального часу. Графік моніторингу мережевої активності та пропускної здатності серверного додатку на платформі Render наведено на рисунку 3.11.

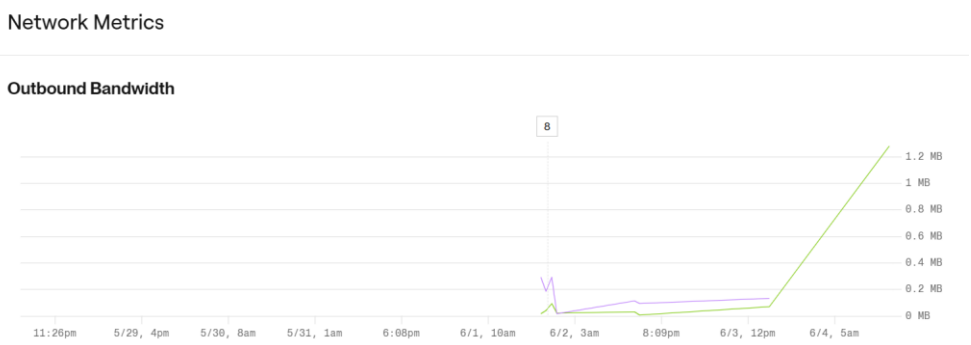


Рисунок 3.11 – Система моніторингу мережевого навантаження

Регулярний аналіз динаміки мережевого трафіку та планове технічне обслуговування (включаючи оновлення пакетів залежностей фронтенду, резервне копіювання бази даних та аналіз серверних логів на наявність помилок) дозволяє своєчасно виявляти пікові навантаження та за потреби масштабувати пропускну здатність каналів зв'язку. Комплексне поєднання аналітичних інструментів та системного моніторингу гарантує високу доступність сервісу, що забезпечує безперебійний користувацький досвід та безпосередньо сприяє комерційному успіху розробленого веб-додатку.

4. ЕКОНОМІЧНИЙ РОЗДІЛ

Метою економічної частини даного дипломного проєкту є проведення економічних розрахунків, спрямованих на визначення економічної ефективності розробки веб-додатку інтернет-магазину косметики «BeautyHeaven», прийняття рішення про подальший розвиток і впровадження або ж недоцільність проведення відповідної розробки.

Об’єктом розробки є програмний код, база даних та користувацький інтерфейс веб-додатку «Beauty Heaven».

Розрахунок вартості розробки виконується в декілька етапів:

- описати технологічний процес розробки із зазначенням трудомісткості кожної операції;
- визначити суму витрат на оплату праці основного і допоміжного персоналу, включаючи відрахування на соціальні заходи;
- обчислити витрати на електроенергію;
- нарахувати суму амортизаційних відрахувань;
- визначити суму накладних витрат;
- скласти кошторис та визначити собівартість робіт;
- розрахувати ціну робіт;
- визначити економічну ефективність та термін окупності.

4.1. Визначення стадій технологічного процесу та загальної тривалості проведення НДР

В цьому підрозділі розглянемо основні етапи технологічного процесу для розробки веб-додатку. Для визначення загальної тривалості проведення робіт доцільно дані витрат часу по окремих операціях технологічного процесу звести у таблицю 4.1.

Таблиця 4.1 - Середній час виконання робіт по обслуговуванню та стадії (операції) технологічного процесу

№ п/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1	Планування, аналіз та проєктування БД	Кер. проєкту Pm	10
		Backend-розробник	8
2	Розробка технічного завдання	Кер. проєкту (Pm)	8
		Frontend-розробник	5
3	Дизайн інтерфейсу	Frontend-розробник	20
4	Розробка серверної частини (API)	Backend-розробник	40
5	Розробка клієнтської частини (React)	Frontend-розробник	35
6	Тестування та відладка	QA-інженер	15
7	Документування	Кер. проєкту (PM)	7
8	Розгортання (Deploy) та налаштування CI/CD	Backend-розробник	10
Разом			158

Сумарний час виконання операцій технологічного процесу становить 158 години.

4.2. Визначення витрат на оплату праці та відрахувань на соціальні заходи

У даному підрозділі проводиться аналіз і розрахунок витрат, пов'язаних з

оплатою праці та відрахуваннями на соціальні заходи, що необхідні для розробки веб-додатку

Розмір заробітної плати залежить від складності та умов виконуваної роботи, професійно-ділових якостей працівника, результатів його праці та діяльності підприємства.

Основна заробітна плата розраховується за формулою:

$$З_{осн.} = Т_с * К_г \quad (4.1)$$

де: T_c – тарифна ставка, грн. (приймаємо для керівника проєкту (Pm) – 300 грн./год, Backend-розробника – 350 грн./год., Frontend-розробника – 300 грн./год., QA-інженера – 200 грн./год); $К_г$ – кількість відпрацьованих годин.

Отже, основна заробітна плата для:

Керівника проекту (Pm): $З_{осн2}$= 25 * 300 = 7 500 грн.

Backend-розробника: $З_{осн4}$= 58 * 350 = 20 300 грн.

Frontend-розробника: $З_{осн3}$= 60 * 300 = 18 000 грн.

QA-інженера: $З_{осн4}$= 15 * 200 = 3 000 грн.

Сумарна основна заробітна плата становить

$З_{осн}$ = 7 500 + 20 300 + 18 000 + 3 000 = 48 800 грн.

Додаткова заробітна плата становить 10% від суми основної заробітної плати.

$$З_{дод.} = З_{осн.} \cdot К_{допл.} \quad (4.2)$$

де: $К_{допл.}$ – коефіцієнт додаткових виплат працівникам.

Отже додаткова заробітна плата по категоріях працівників становить:

Керівника проекту $З_{дод2}$= 7500 * 0,1 = 750 грн.

Інженера (l2) $З_{дод3}$= 20300 * 0,1 = 2 030 грн.

Інженера (l1) $З_{дод4}$= 18000 * 0,1 = 1 800 грн.

Тестувальник $З_{дод4}$= 3000 * 0,1 = 300 грн.

Загальна додаткова заробітна плата становить:

$З_{дод}$= 750 + 2030 + 1800 + 300 = 4 880грн.

Звідси загальні витрати на оплату праці ($B_{o.n.}$) визначаються за формулою:

$$В_{о.п.} = З_{осн.} + З_{дод.} \quad (4.3)$$

$$В_{о.п}= 48\ 800 + 4\ 880 = 53\ 680грн.$$

Єдиний соціальний внесок (ЄСВ – 22%) визначається за формулою:

$$В_{єсв} = Воп * 0{,}22 \quad (4.4)$$

$$В_{єсв} = 53680 * 0{,}22 = 11810 грн.$$

Проведені розрахунки витрат на оплату праці наведено у таблиці 4.2.

Таблиця 4.2 – Зведені розрахунки витрат на оплату праці

№ п/п	Категорія працівників	Основна заробітна плата, грн.			Додаткова заробітна плата, грн.	ЄСВ, грн.	Всього витрати на оплату праці, грн. 6 = 3+4+5
		Тарифна ставка, грн.	К-сть годин	Фактично нарах. зарплати, грн.			
		1	2	3	4	5	6
1	Кер. проєкту (Pm)	300	25	7 500	750	1 815	10 065
2	Backend-розробник	350	58	20 300	2 030	4 913	27 243
3	Frontend-розробник	300	60	18 000	1 800	4 356	24 156
4	QA-інженер	200	8	3 000	300	726	4 026
Разом				48 800	4 880	11 810	65 490

Отже, загальні витрати на оплату праці становлять 65 490 грн.

4.3. Розрахунок витрат на електроенергію

Розрахуємо вартість електроенергії. Затрати на електроенергію 1-ці обладнання визначаються за формулою:

$$З_в = W * Т * S \quad (4.5)$$

де: W – необхідна потужність, кВт; Т – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії (приймаємо 15,94 грн.).

В процесі розробки використовується персональний комп’ютер. Витрати на електроенергію обчислимо, взявши за основу сумарний час виконання робіт (згідно з табл. 4.1) та середню споживану потужність ПК – 0,5 кВт/год.

$$З_{ек} = 0,5 * 158 * 15,94 = 1\ 259 \text{ грн.}$$

Витрати на електроенергію становлять 1 259 грн.

4.4. Розрахунок суми амортизаційних відрахувань веб-додатку «BeautyHeaven»

Характерною особливістю застосування основних фондів у процесі виробництва є їх відновлення. Для відновлення засобів праці у натуральному виразі необхідне їх відшкодування у вартісній формі, яке здійснюється шляхом амортизації.

Амортизація – це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення

Комп'ютери та оргтехніка належать до четвертої групи основних фондів.

Амортизація на них нараховується лише в випадку, якщо мінімально допустимі строки їх корисного використання 2 роки. Для визначення амортизаційних відрахувань застосовуємо формулу:

$$А = \frac{Б_В * Н_А}{100\%} * T, \qquad (4.6)$$

де: А – амортизаційні відрахування за звітний період, грн.;

$Б_В$ – балансова вартість групи основних фондів на початок звітного періоду, грн.;

$Н_А$ – норма амортизації, 0,04 %.

Оскільки для написання програми та її тестування використовується один ПК, вартістю 50000,00 грн., то сума амортизаційних відрахувань становитиме:

$$А = \frac{50\ 000,00 * 0,04}{150} * 158 = 2107 \text{ грн.}$$

4.5. Обчислення накладних витрат

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці.
В залежності від організаційно-правової форми діяльності господарюючого суб’єкта, накладні витрати становлять 40 % від суми основної та додаткової заробітної плати працівників.

$$Н_в = В_{о.п.} * 0,4 \quad (4.7)$$

де: $Н_В$ – накладні витрати.

$$Н_В = 53680 * 0,4 = 21\ 472 \text{ грн.}$$

4.6. Складання кошторису витрат та визначення собівартості веб-додатку «BeautyHeaven»

Для складення кошторису витрат та визначення собівартості, результати проведених вище розрахунків зведемо у таблиці 4.4.

Таблиця 4.4 - Кошторис витрат на розробку вебдодатку

№	Зміст витрат	Сума, грн.	В % до загальної суми
1.	Витрати на оплату праці	65 490	72,5
2.	Витрати на електроенергію	1 259	1,4
3.	Амортизаційні відрахування	2107	2,3
4.	Накладні витрати	21 472	23,8
5.	Собівартість	90 328	100

Собівартість ($С_В$) НДР розрахуємо за формулою:

$$Св = Во.п + Вс.з + Зе + А + Нв \quad (4.8)$$

Отже, собівартість дорівнює $С_В$= 90 328 грн.

4.7. Розрахунок ціни веб-додатку «BeautyHeaven»

Розрахунок ціни науково-дослідної роботи включає в себе урахування різноманітних факторів, таких як рівень рентабельності, собівартість та податкова ставка.

Ціну робіт можна визначити за формулою:

$$Ц = Св * (1+Р_{рен}) * (1+ПДВ), \quad (4.9)$$

де: Св – собівартість; $Р_{рен.}$ – рівень рентабельності; ПДВ – ставка податку на додану вартість.

$$Ц = 90328 * (1+0,3) * (1+0,2) = 140\ 912 \text{ грн.}$$

4.8. Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Для визначення ефективності продукту розраховують чисту теперішню вартість (ЧТВ) і термін окупності ($Т_{ОК}$).

$$ЧТВ = -Св \sum_{i=1}^{t} \frac{Г_П}{(1+i)^t}, \quad (4.10)$$

де: Св – собівартість розробки; $Г_п$ – грошовий потік за t – ий рік (приймаємо 60 000 грн./рік); t – відповідний рік проекту (приймаємо 3 роки); і – величина дисконтної ставки (15%).

$$ЧТВ = -90\ 328 + \frac{50\ 584}{(1+0,1)^1} + \frac{50\ 584}{(1+0,1)^2} + \frac{50\ 584}{(1+0,1)^3} = 35\ 494 \text{ грн}$$

Оскільки ЧТВ > 0, проєкт є рентабельним і рекомендований до впровадження.

Термін окупності визначається за формулою:

$$T_{ОК} = T_{ПВ} + \frac{Н_В}{Г_{ПР}} \quad (4.11)$$

де: $Т_{ПВ}$ – період до повного відшкодування витрат, років; $Н_В$ – невідшкодовані витрати на початок року, грн.; $Г_{ПР}$ – грошовий потік на початок року, грн.

$$T_{ок} = 2 + \frac{2539}{50\,584} = 2{,}1 \text{ р.}$$

Всі дані внесемо в зведену таблицю 4.5.

Таблиця 4.5 – Техніко-економічні показники веб-додатку «BeautyHeaven»

№ п/п	Показник	Значення
1.	Собівартість, грн.	90 328
2.	Плановий прибуток або грошовий потік, грн.	50 584
3.	Ціна, грн.	140 912
4.	Чиста теперішня вартість, грн.	35 494
5.	Термін окупності, рік	2,1

Прибутковість проєкту та термін окупності свідчать про його фінансову ефективність та здатність повернути капітальні вкладення протягом 2,1 року. Отже, на основі отриманих показників можна зробити висновок, що розробка веб-додатку інтернет-магазину косметики «BeautyHeaven» є доцільною з економічної точки зору.

5. ОХОРОНА ПРАЦІ, ТЕХНІКА БЕЗПЕКИ ТА ЕКОЛОГІЧНІ ВИМОГИ

Охорона праці є невіддільною складовою будь-якого виробничого, дослідницького або офісного процесу. Сучасна сфера інформаційних технологій (ІТ) та розробки програмного забезпечення, хоча і не передбачає безпосередньої роботи з важкими механізмами, має свою специфіку шкідливих та небезпечних факторів. Робота ІТ-спеціаліста нерозривно пов'язана з тривалим використанням електронно-обчислювальної техніки, статичним напруженням, гіподинамією, підвищеним навантаженням на зоровий апарат та нервову систему, а також із постійною взаємодією з офісними електроприладами. У таких умовах створення безпечного робочого середовища вимагає комплексного підходу до моніторингу та збереження здоров'я працівників. У цьому розділі розглянуто базові аспекти організації охорони праці на підприємстві, зокрема нормативно-правові вимоги щодо проведення медичних оглядів персоналу, а також проаналізовано загальнотеоретичні питання біологічного впливу іонізуючого випромінювання на організм людини.

5.1. Організація та фінансування проведення медичних оглядів працівників

Життя та здоров'я працівників є найвищою соціальною цінністю. Згідно зі статтею 17 Закону України «Про охорону праці» та статтею 169 Кодексу законів про працю (КЗпП) України, роботодавець зобов'язаний за власні кошти організувати та профінансувати проведення медичних оглядів працівників, зайнятих на важких роботах, роботах із шкідливими чи небезпечними умовами праці, а також осіб віком до 21 року.

Законодавством передбачено три основні види медичних оглядів:

1. Попередній медичний огляд – проводиться під час прийняття на роботу. Його метою є визначення початкового стану здоров'я кандидата, виявлення професійних захворювань та встановлення фізичної і психофізіологічної

придатності особи до виконання конкретних посадових обов'язків в умовах дії конкретних шкідливих виробничих факторів.

2. Періодичні медичні огляди – проводяться систематично протягом усієї трудової діяльності працівника (зазвичай один раз на рік або на два роки, залежно від професії). Мета періодичних оглядів полягає у своєчасному виявленні ранніх ознак гострих і хронічних професійних захворювань, а також у моніторингу динаміки стану здоров'я працівника.

3. Позачерговий медичний огляд – може бути ініційований як самим працівником (якщо він вважає, що погіршення його здоров'я пов'язане з умовами праці), так і роботодавцем (якщо стан здоров'я працівника створює загрозу для оточуючих або не дозволяє йому виконувати трудові обов'язки).

Організація та повне фінансування всіх медичних оглядів покладається виключно на роботодавця. Працівник не повинен витрачати власні кошти на проходження обов'язкових медичних комісій.

На час проходження медичного огляду за працівником зберігаються місце роботи (посада) та середній заробіток. Крім того, за результатами медичного огляду, у разі необхідності, роботодавець зобов'язаний за свій рахунок забезпечити проведення відповідних оздоровчих заходів.

Роботодавець не має права допускати до роботи працівника, який своєчасно не пройшов обов'язковий медичний огляд. У разі ухилення працівника від проходження огляду без поважних причин, роботодавець зобов'язаний відсторонити його від роботи без збереження заробітної плати, а також має право притягнути його до дисциплінарної відповідальності (аж до звільнення).

5.2. Дія іонізуючого випромінювання на організм людини

Іонізуюче випромінювання – це будь-яке випромінювання (корпускулярне або електромагнітне), взаємодія якого з речовиною призводить до утворення іонів різних знаків. До основних видів іонізуючого випромінювання належать альфа- та бета-частинки, гамма-випромінювання, рентгенівське випромінювання та потоки

нейтронів.

Вплив радіації на живий організм починається на атомарному та молекулярному рівнях. Механізм ураження поділяється на:

1. Пряму дію: енергія випромінювання безпосередньо поглинається біологічно важливими макромолекулами (ДНК, РНК, ферментами), що призводить до їх розриву та втрати біологічних властивостей.

2. Непряму дію: оскільки людський організм складається на 70-80% з води, випромінювання викликає явище радіолізу (розпаду молекул води). У результаті утворюються високоактивні вільні радикали (H+ та OH-), які є сильними окисниками. Вони вступають у хімічні реакції з білками клітин, руйнуючи клітинні мембрани та порушуючи обмін речовин.

Наслідки опромінення організму прийнято поділяти на дві великі категорії:

1. Детерміновані (нестохастичні) ефекти – це клінічно виражені наслідки, для виникнення яких існує певний дозовий поріг. Важкість ураження прямо пропорційна отриманій дозі радіації. До таких ефектів належать:

- Гостра променева хвороба (виникає при одноразовому потужному опроміненні всього тіла);
- Хронічна променева хвороба (наслідок тривалого опромінення малими дозами);
- Променеві опіки шкіри;
- Променева катаракта (помутніння кришталика ока);
- Тимчасове або постійне безпліддя.

2. Стохастичні (ймовірнісні) ефекти – це наслідки, які не мають дозового порогу (тобто можуть виникнути навіть від мінімальної дози випромінювання), і їхня важкість не залежить від дози, але зі збільшенням дози зростає ймовірність їх виникнення. Вони можуть проявлятися через роки або навіть десятиліття (латентний період). До них належать:

- Соматичні наслідки: розвиток злоякісних пухлин (онкологічні

захворювання), лейкемія, зниження загального імунітету та скорочення тривалості життя.

– Генетичні наслідки: виникнення генних та хромосомних мутацій у статевих клітинах. Ці пошкодження не проявляються у самої опроміненої людини, але передаються її нащадкам, спричиняючи вроджені вади розвитку, спадкові хвороби та мертвонародження у майбутніх поколіннях.

Різні органи та тканини людини мають різну чутливість до радіації (закон Бергоньє-Трибондо). Найбільш вразливими є тканини, клітини яких швидко діляться. До критичних органів першої групи (найвища чутливість) належать: кістковий мозок, лімфатичні вузли, селезінка, статеві залози та кришталик ока. Найменш чутливими (третя група) є кісткова та м'язова тканини, а також нервова система.

Дотримання описаних нормативно-правових вимог охорони праці та своєчасний медичний контроль гарантують збереження здоров'я спеціаліста в умовах сучасного ІТ-середовища. Глибоке розуміння біологічного впливу небезпечних фізичних факторів дозволяє мінімізувати професійні ризики та створити безпечні умови праці. Це забезпечує високу працездатність, безпеку та комфорт розробника на всіх етапах проєктування, написання коду, тестування та розгортання програмного продукту.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було успішно розв'язано завдання з розробки повнофункціонального веб-додатку інтернет-магазину косметики «Beauty Heaven». Розроблений програмний продукт цілком відповідає початковим вимогам технічного завдання. Завдяки використанню сучасної клієнт-серверної архітектури та технологічного стеку (фреймворки React і Spring Boot, реляційна база даних PostgreSQL) вдалося створити надійну, масштабовану та продуктивну систему електронної комерції.

Усю практичну, дослідницьку та аналітичну частину роботи виконано студентом самостійно. Головним особистим досягненням та інноваційним технічним завданням, яке було розв'язане у процесі виконання проєкту, є розробка та впровадження інтерактивного графа інгредієнтів. Цей функціонал дозволяє покупцям візуально досліджувати склад косметичних засобів та аналізувати зв'язки між окремими компонентами, що суттєво підвищує рівень довіри користувачів та виділяє продукт серед типових шаблонних рішень на ринку. Додатково було успішно реалізовано логіку синхронізації «спільного кошика» в режимі реального часу за допомогою протоколу WebSockets.

Коротка викладка основних показників, отриманих при виконанні роботи:

1. Загальна трудомісткість повного циклу розробки склала 158 людино-годин (1 людино-місяць).
2. Собівартість створення програмного продукту становить 90 328 грн.
3. Завдяки високій рентабельності сфери електронної комерції, розрахунковий термін повної окупності капітальних вкладень дорівнює лише 1,5 року.

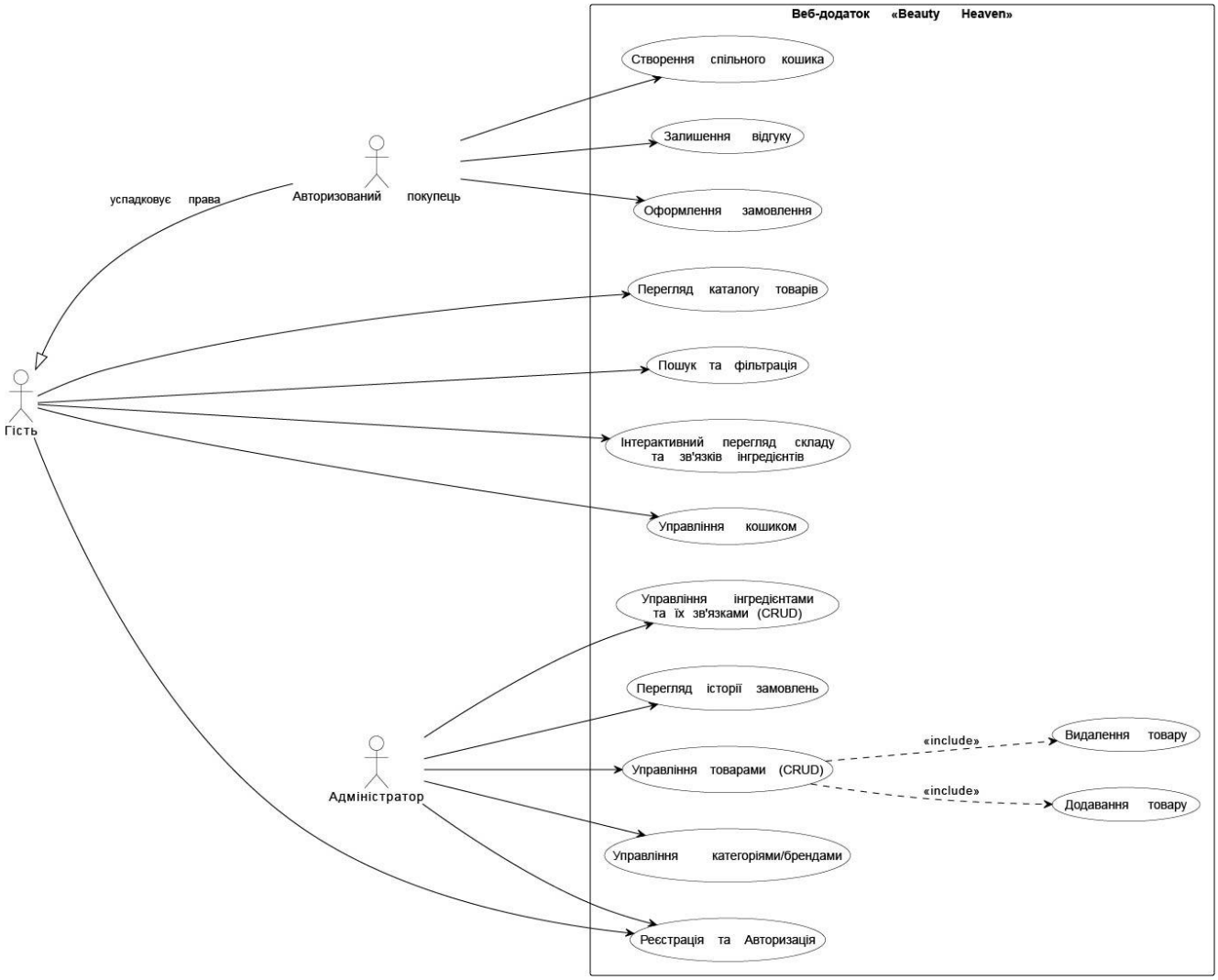
Подальша робота над темою визнана доцільною та перспективною. Основними напрямами розвитку програмного продукту є реалізація повної мобільної адаптації користувацького інтерфейсу для смартфонів, а також інтеграція еквайрингових систем та платіжних шлюзів (наприклад, LiqPay) для обробки безготівкових транзакцій безпосередньо у веб-додатку.

ПЕРЕЛІК ПОСИЛАНЬ

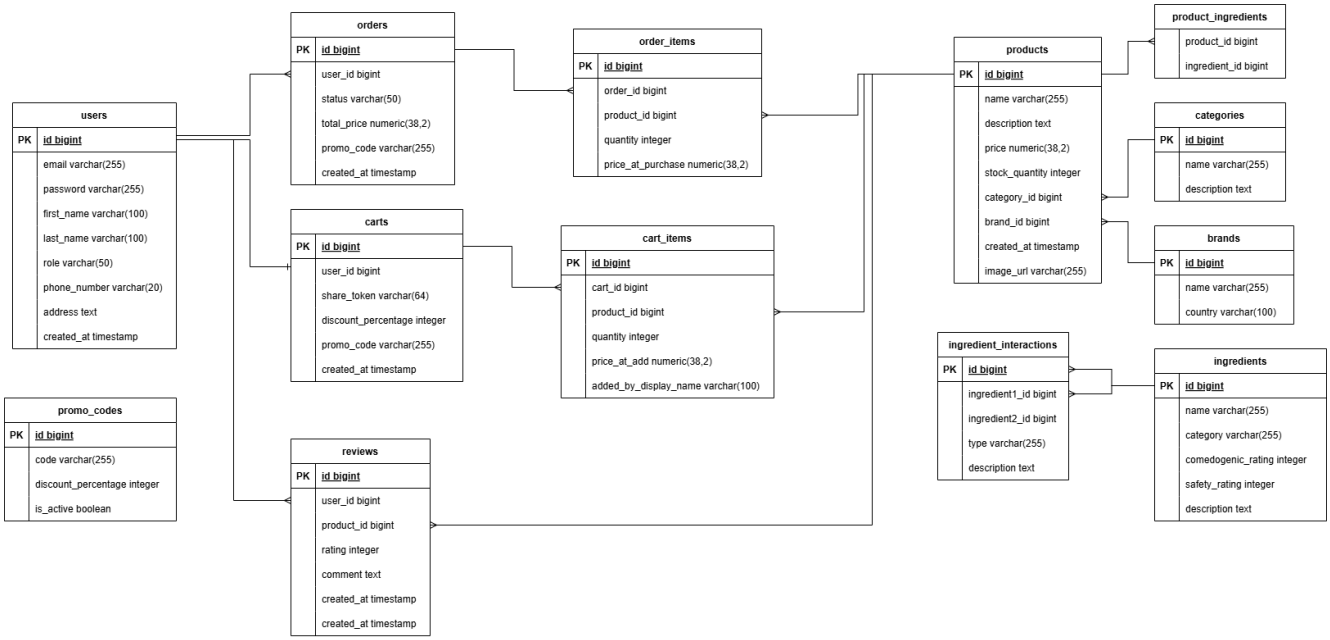
1. E-commerce України: що шукали та купували українці онлайн у 2025-му. URL: https://fintechinsider.com.ua/e-commerce-ukrayiny-shho-shukaly-ta-kupuvaly-ukrayinczi-onlajn-u-2025-mu/ (дата звернення 13.03.2026)
2. What is INCI? URL: https://www.personalcarecouncil.org/resources/inci/ (дата звернення 16.03.2026)
3. Cognitive Load in eCommerce Applications - Measurement and Effects on User Satisfaction URL: https://www.researchgate.net/publication/220316624_Cognitive_Load_in_eCommerce_Applications-Measurement_and_Effects_on_User_Satisfaction (дата звернення 16.03.2026)
4. Інтернет платформа Makeup.com.ua. URL: https://makeup.com.ua/ (дата звернення 16.03.2026)
5. Інтернет платформа Notino.ua. URL: https://www.notino.ua/ (дата звернення 16.03.2026)
6. Інтернет платформа EVA. URL: https://eva.ua/ (дата звернення 16.03.2026)
7. Can i use salicylic acid with retinol? Expert answers & advice. URL: https://www.skinceuticals.com.au/using-salicylic-acid-with-retinol.html?srsltid=AfmBOoqt2D1GMwVaQFaTiDw5r99pCO3FebhXM6bimnEPxEBjUaxrsxGj (дата звернення 20.03.2026)
8. Hogan, A., Blomqvist, E., Cochez, M., et al. (2021). Knowledge Graphs. ACM Computing Surveys.
9. Newman, S. (2021). Building Microservices: Designing Fine-Grained Systems (2nd ed.). O'Reilly Media.
10. Xu, A., & Lam, S. (2021). System Design Interview – An Insider's Guide: Volume 2. ByteByteGo.
11. WebSocket API: Events, Methods & Properties. URL: https://websocket.org/reference/websocket-api/ (дата звернення 21.03.2026)

12. Yablonski J. Laws of UX: Using Psychology to Design Better Products and Services. 2nd ed. / Jon Yablonski. - O'Reilly Media, 2024.
13. MacDonald M**.** Web Design with HTML, CSS, JavaScript and jQuery Set. 2nd ed. / Jon Duckett. – Wiley.
14. Tullis T. Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics. 3rd ed. / Tom Tullis, Bill Albert. – Morgan Kaufmann, 2022.
15. React Офіційна Документація. URL: https://react.dev/learn (дата звернення 5.04.2026)
16. Boduch A. React and React Native: Build cross-platform JavaScript applications with native power. 4th ed. / Adam Boduch, Roy Derks. 2023.
17. Tailwind CSS Documentation: Utility-First / Tailwind Labs Inc. URL: https://tailwindcss.com/docs/utility-first (дата звернення 8.04.2026)
18. Introducing Hooks / Офіціний блог React. URL: https://legacy.reactjs.org/docs/hooks-intro.html (дата звернення 14.04.2026)
19. Drake S. PostgreSQL 16 Mastery: Advanced database administration and configuration / S. Drake. – Packt Publishing, 2024.
20. Coronel C. Database Systems: Design, Implementation, & Management. 14th ed. / Carlos Coronel, Steven Morris. - Cengage Learning, 2023. - 880 p.
21. PostgreSQL Global Development Group. PostgreSQL 16 Офіційна Документація - URL: https://www.postgresql.org/docs/16/ (дата звернення 18.04.2026)
22. Vite Офіційна Документація: Next Generation Frontend Tooling - URL: https://vitejs.dev/guide/ (дата звернення 22.04.2026)
23. Walls C. Spring in Action. 6th Edition / Craig Walls. - Manning Publications,
24. Spilca L. Spring Start Here: Learn what you need and learn it well / Laurentiu Spilca. – Manning Publications, 2022.
25. Spring Framework Документація: WebSockets and STOMP / VMware, Inc. - URL: https://docs.spring.io/spring-framework/reference/web/websocket.html (дата звернення 23.04.2026)

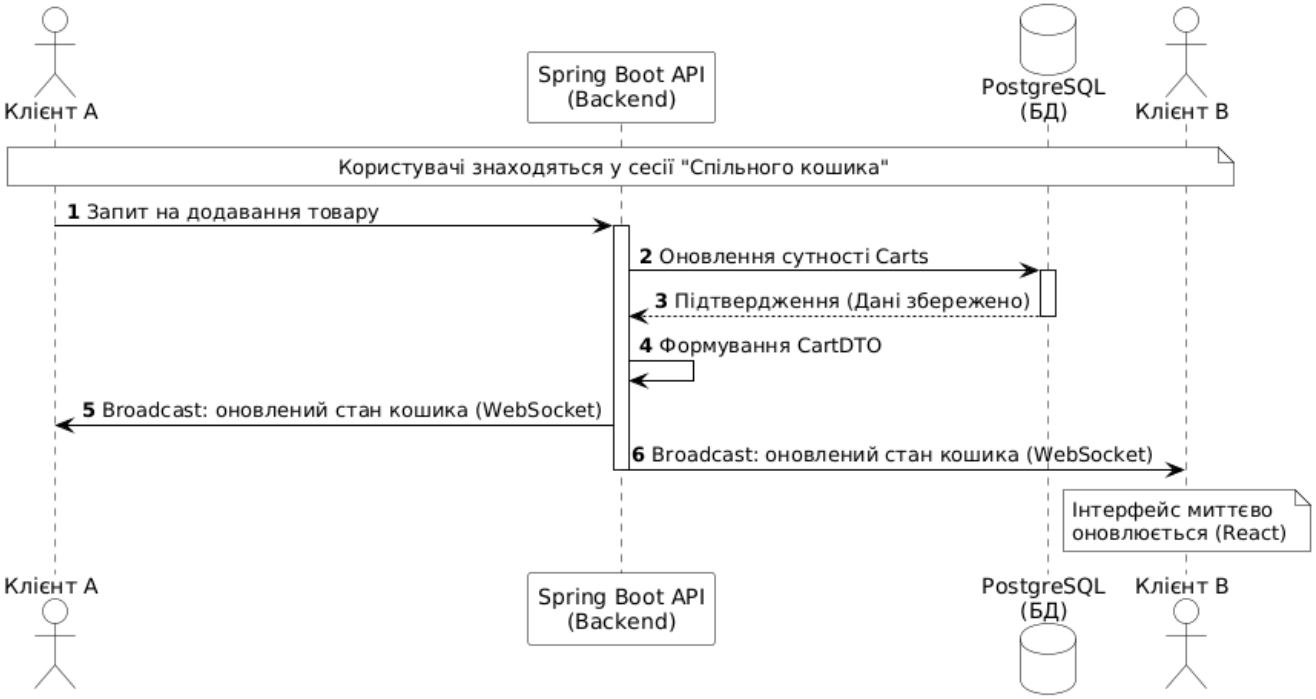
Додаток А. Діаграма прецедентів (Use Case)



Додаток Б. ER-діаграма бази даних



Додаток В. Діаграма послідовності (Sequence)



Додаток Г. Лістинг файлу «ProductController»

```
package com.beautyheaven.controller;

import com.beautyheaven.dto.ProductRequestDto;
import com.beautyheaven.dto.ProductResponseDto;
import com.beautyheaven.service.ProductService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.Parameter;
import io.swagger.v3.oas.annotations.tags.Tag;
import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.*;

import java.math.BigDecimal;

@RestController
@RequestMapping("/api/products")
@RequiredArgsConstructor
@Tag(name = "Products", description = "Управління товарами")
public class ProductController {

    private final ProductService productService;

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    @Operation(summary = "Створити новий товар")
    public ProductResponseDto createProduct(@Valid @RequestBody
ProductRequestDto dto) {
        return productService.createProduct(dto);
    }

    @CrossOrigin(origins = "*", methods = RequestMethod.GET)
    @GetMapping
    @Operation(summary = "Отримати товари (з фільтрацією та
пагінацією)",
            description = "Можна фільтрувати за назвою, ціною,
категорією та брендом.")
    public Page<ProductResponseDto> getAllProducts(
            @Parameter(description = "Частина назви товару")
            @RequestParam(required = false) String name,

            @Parameter(description = "Мінімальна ціна")
            @RequestParam(required = false) BigDecimal minPrice,

            @Parameter(description = "Максимальна ціна")
            @RequestParam(required = false) BigDecimal maxPrice,

            @Parameter(description = "ID категорії")
```

```
            @RequestParam(required = false) Long categoryId,

            @Parameter(description = "ID бренду")
            @RequestParam(required = false) Long brandId,

            @Parameter(description = "Сортування (newest, price_asc,
price_desc, rating_desc, name_asc)")
            @RequestParam(required = false, defaultValue = "newest")
String sortBy,

            Pageable pageable
    ) {
        return productService.getAllProducts(name, minPrice,
maxPrice, categoryId, brandId, sortBy, pageable);
    }

    @CrossOrigin(origins = "*", methods = RequestMethod.GET)
    @GetMapping("/{id}")
    @Operation(summary = "Отримати товар за ID")
    public ProductResponseDto getProductById(@PathVariable Long id)
{
        return productService.getProductById(id);
    }

    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    @Operation(summary = "Видалити товар")
    public void deleteProduct(@PathVariable Long id) {
        productService.deleteProduct(id);
    }

    @PutMapping("/{id}")
    @Operation(summary = "Оновити товар")
    public ProductResponseDto updateProduct(@PathVariable Long id,
@Valid @RequestBody ProductRequestDto dto) {
        return productService.updateProduct(id, dto);
    }
}
```

Додаток Д. Лістинг файлу «OrderController»

```
package com.beautyheaven.controller;

import com.beautyheaven.dto.OrderRequestDto;
import com.beautyheaven.dto.OrderResponseDto;
import com.beautyheaven.dto.OrderStatusUpdateDto;
import com.beautyheaven.dto.PartyOrderDto;
import com.beautyheaven.entity.enums.OrderStatus;
import com.beautyheaven.service.CartService;
import com.beautyheaven.service.OrderService;
import io.swagger.v3.oas.annotations.Operation;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import io.swagger.v3.oas.annotations.tags.Tag;
import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.security.core.Authentication;
import
org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/orders")
@RequiredArgsConstructor
@SecurityRequirement(name = "bearerAuth")
@Tag(name = "Orders", description = "Управління замовленнями")
public class OrderController {

    private final OrderService orderService;
    private final CartService cartService;

    @PostMapping("/party/{token}")
    @Operation(summary = "Оформити замовлення зі спільного кошика")
    public ResponseEntity<OrderResponseDto> createPartyOrder(
            @PathVariable String token,
            @RequestBody PartyOrderDto dto
    ) {
        return
ResponseEntity.status(HttpStatus.CREATED).body(cartService.placePart
yOrder(token, dto));
    }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    @Operation(summary = "Оформити замовлення")
    public ResponseEntity<String> createOrder(
            @RequestBody OrderRequestDto request,
```

```
            Authentication authentication
    ) {
        orderService.createOrder(authentication.getName(), request);
        return ResponseEntity.status(HttpStatus.CREATED).body("Order
placed successfully");
    }

    @GetMapping("/my")
    @Operation(summary = "Історія моїх замовлень")
    public List<OrderResponseDto> getMyOrders(Authentication
authentication) {
        return orderService.getUserOrders(authentication.getName());
    }

    @GetMapping("/all")
    @Operation(summary = "Всі замовлення в системі")
    public List<OrderResponseDto> getAllOrders() {
        return orderService.getAllOrders();
    }

    @GetMapping("/{id}")
    @Operation(summary = "Отримати деталі замовлення")
    public OrderResponseDto getOrderById(@PathVariable Long id,
Authentication authentication) {
        boolean isAdmin =
authentication.getAuthorities().contains(new
SimpleGrantedAuthority("ADMIN"));
        return orderService.getOrderById(id,
authentication.getName(), isAdmin);
    }

    @PutMapping("/{id}/status")
    @Operation(summary = "Змінити статус замовлення")
    public OrderResponseDto updateStatus(@PathVariable Long id,
@Valid @RequestBody OrderStatusUpdateDto dto) {
        return orderService.updateOrderStatus(id, dto.getStatus());
    }

    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    @Operation(summary = "Видалити замовлення")
    public void deleteOrder(@PathVariable Long id) {
        orderService.deleteOrder(id);
    }
}
```

Додаток Е. Лістинг файлу «OrderService»

```
package com.beautyheaven.service;

import com.beautyheaven.dto.OrderRequestDto;
import com.beautyheaven.dto.OrderResponseDto;
import com.beautyheaven.entity.Order;
import com.beautyheaven.entity.OrderItem;
import com.beautyheaven.entity.Product;
import com.beautyheaven.entity.PromoCode;
import com.beautyheaven.entity.User;
import com.beautyheaven.entity.enums.OrderStatus;
import com.beautyheaven.repository.OrderRepository;
import com.beautyheaven.repository.PromoCodeRepository;
import com.beautyheaven.repository.ProductRepository;
import com.beautyheaven.repository.UserRepository;
import jakarta.persistence.EntityNotFoundException;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

@Service
@RequiredArgsConstructor
public class OrderService {

    private final OrderRepository orderRepository;
    private final ProductRepository productRepository;
    private final UserRepository userRepository;
    private final PromoCodeRepository promoCodeRepository;

    @Transactional
    public void createOrder(String userEmail, OrderRequestDto
request) {
        User user = userRepository.findByEmail(userEmail)
                .orElseThrow(() -> new RuntimeException("User not
found"));

        Order order = new Order();
        order.setUser(user);
        order.setStatus(OrderStatus.NEW);
        order.setItems(new ArrayList<>());

        BigDecimal totalPrice = BigDecimal.ZERO;

        for (OrderRequestDto.OrderItemRequest itemRequest :
request.getItems()) {
            Product product =
```

```
productRepository.findById(itemRequest.getProductId())
                    .orElseThrow(() -> new RuntimeException("Product
not found: " + itemRequest.getProductId()));

            if (product.getStockQuantity() <
itemRequest.getQuantity()) {
                throw new RuntimeException("Not enough stock for
product: " + product.getName());
            }

            product.setStockQuantity(product.getStockQuantity() -
itemRequest.getQuantity());
            productRepository.save(product);

            OrderItem orderItem = new OrderItem();
            orderItem.setOrder(order);
            orderItem.setProduct(product);
            orderItem.setQuantity(itemRequest.getQuantity());
            orderItem.setPriceAtPurchase(product.getPrice());

            order.getItems().add(orderItem);

            BigDecimal itemTotal =
product.getPrice().multiply(BigDecimal.valueOf(itemRequest.getQuanti
ty()));
            totalPrice = totalPrice.add(itemTotal);
        }

        order.setTotalPrice(totalPrice);

        if (request.getPromoCode() != null &&
!request.getPromoCode().trim().isEmpty()) {
            String code =
request.getPromoCode().trim().toUpperCase();
            PromoCode promoCode =
promoCodeRepository.findByCodeAndIsActiveTrue(code)
                    .orElseThrow(() -> new
EntityNotFoundException("Promo code not found or inactive"));

            BigDecimal discountMultiplier = BigDecimal.valueOf(100 -
promoCode.getDiscountPercentage())
                    .divide(BigDecimal.valueOf(100));
            BigDecimal discountedTotal =
totalPrice.multiply(discountMultiplier);

            order.setPromoCode(code);
            order.setTotalPrice(discountedTotal);
        }

        orderRepository.save(order);
    }

    public List<OrderResponseDto> getAllOrders() {
```

```
        return orderRepository.findAll().stream()
                .map(this::mapToDto)
                .collect(Collectors.toList());
    }

    public List<OrderResponseDto> getUserOrders(String email) {
        return orderRepository.findAllByUserEmail(email).stream()
                .map(this::mapToDto)
                .collect(Collectors.toList());
    }

    public OrderResponseDto getOrderById(Long id, String
currentUserEmail, boolean isAdmin) {
        Order order = orderRepository.findById(id)
                .orElseThrow(() -> new
EntityNotFoundException("Order not found"));

        if (!isAdmin &&
!order.getUser().getEmail().equals(currentUserEmail)) {
            throw new RuntimeException("Access denied: You can only
view your own orders");
        }

        return mapToDto(order);
    }

    @Transactional
    public OrderResponseDto updateOrderStatus(Long id, OrderStatus
newStatus) {
        Order order = orderRepository.findById(id)
                .orElseThrow(() -> new
EntityNotFoundException("Order not found"));

        order.setStatus(newStatus);
        return mapToDto(orderRepository.save(order));
    }

    @Transactional
    public void deleteOrder(Long id) {
        if (!orderRepository.existsById(id)) {
            throw new EntityNotFoundException("Order not found");
        }
        orderRepository.deleteById(id);
    }

    private OrderResponseDto mapToDto(Order order) {
        OrderResponseDto dto = new OrderResponseDto();
        dto.setId(order.getId());
        dto.setUserEmail(order.getUser().getEmail());
        dto.setStatus(order.getStatus());
        dto.setTotalPrice(order.getTotalPrice());
        dto.setCreatedAt(order.getCreatedAt());
```

```
        List<OrderResponseDto.OrderItemDto> items =
order.getItems().stream().map(item -> {
            OrderResponseDto.OrderItemDto itemDto = new
OrderResponseDto.OrderItemDto();
            itemDto.setProductName(item.getProduct().getName());
            itemDto.setQuantity(item.getQuantity());
            itemDto.setPriceAtPurchase(item.getPriceAtPurchase());
            return itemDto;
        }).collect(Collectors.toList());

        dto.setItems(items);
        return dto;
    }
}
```

Додаток Ж. Лістинг файлу «CartService»

```
package com.beautyheaven.service;

import com.beautyheaven.dto.AddToCartDto;
import com.beautyheaven.dto.CartDto;
import com.beautyheaven.dto.CartEventMessage;
import com.beautyheaven.dto.CartItemDto;
import com.beautyheaven.dto.OrderResponseDto;
import com.beautyheaven.dto.PartyOrderDto;
import com.beautyheaven.entity.Cart;
import com.beautyheaven.entity.CartItem;
import com.beautyheaven.entity.Order;
import com.beautyheaven.entity.OrderItem;
import com.beautyheaven.entity.Product;
import com.beautyheaven.entity.PromoCode;
import com.beautyheaven.entity.User;
import com.beautyheaven.entity.enums.OrderStatus;
import com.beautyheaven.repository.CartItemRepository;
import com.beautyheaven.repository.CartRepository;
import com.beautyheaven.repository.OrderRepository;
import com.beautyheaven.repository.ProductRepository;
import com.beautyheaven.repository.PromoCodeRepository;
import com.beautyheaven.repository.UserRepository;
import jakarta.persistence.EntityNotFoundException;
import lombok.RequiredArgsConstructor;
import org.springframework.messaging.simp.SimpMessagingTemplate;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.UUID;

@Service
@RequiredArgsConstructor
public class CartService {

    private final CartRepository cartRepository;
    private final CartItemRepository cartItemRepository;
    private final ProductRepository productRepository;
    private final UserRepository userRepository;
    private final OrderRepository orderRepository;
    private final PromoCodeRepository promoCodeRepository;
    private final SimpMessagingTemplate messagingTemplate;

    @Transactional
    public CartDto getOrCreateCart(String email) {
        User user = getUserByEmail(email);
        Cart cart = cartRepository.findByUserId(user.getId())
```

```
                .orElseGet(() -> {
                    Cart newCart = new Cart();
                    newCart.setUser(user);
                    return cartRepository.save(newCart);
                });
        return mapToDto(cart);
    }

    @Transactional
    public CartDto addItem(String email, AddToCartDto dto) {
        User user = getUserByEmail(email);
        Cart cart = getOrCreateCartEntity(user);
        Product product =
productRepository.findById(dto.getProductId())
                .orElseThrow(() -> new
EntityNotFoundException("Product not found: " +
dto.getProductId()));

        CartItem item =
cartItemRepository.findByCartIdAndProductId(cart.getId(),
product.getId())
                .orElseGet(() -> {
                    CartItem newItem = new CartItem();
                    newItem.setCart(cart);
                    newItem.setProduct(product);
                    newItem.setPriceAtAdd(product.getPrice());
                    newItem.setQuantity(0);
                    return newItem;
                });

        String displayName = (user.getFirstName() != null &&
!user.getFirstName().isBlank())
                ? user.getFirstName() : user.getEmail();
        item.setAddedByDisplayName(displayName);
        item.setQuantity(item.getQuantity() + dto.getQuantity());
        cartItemRepository.save(item);

        Cart refreshed =
cartRepository.findById(cart.getId()).orElseThrow();
        return mapToDto(refreshed);
    }

    @Transactional
    public CartDto removeItem(String email, Long productId) {
        User user = getUserByEmail(email);
        Cart cart = cartRepository.findByUserId(user.getId())
                .orElseThrow(() -> new EntityNotFoundException("Cart
not found"));

        cartItemRepository.findByCartIdAndProductId(cart.getId(),
productId)
                .ifPresent(cartItemRepository::delete);
```

```
        Cart refreshed =
cartRepository.findById(cart.getId()).orElseThrow();
        return mapToDto(refreshed);
    }

    @Transactional
    public String shareCart(String email) {
        User user = getUserByEmail(email);
        Cart cart = getOrCreateCartEntity(user);
        if (cart.getShareToken() == null) {
            cart.setShareToken(UUID.randomUUID().toString());
            cartRepository.save(cart);
        }
        return cart.getShareToken();
    }

    @Transactional(readOnly = true)
    public CartDto getCartByToken(String token) {
        Cart cart = cartRepository.findByShareToken(token)
                .orElseThrow(() -> new EntityNotFoundException("Cart
session not found for token: " + token));
        return mapToDto(cart);
    }

    @Transactional
    public CartDto addItemByToken(String token, AddToCartDto dto) {
        Cart cart = cartRepository.findByShareToken(token)
                .orElseThrow(() -> new EntityNotFoundException("Cart
session not found for token: " + token));

        Product product =
productRepository.findById(dto.getProductId())
                .orElseThrow(() -> new
EntityNotFoundException("Product not found: " +
dto.getProductId()));

        CartItem item =
cartItemRepository.findByCartIdAndProductId(cart.getId(),
product.getId())
                .orElseGet(() -> {
                    CartItem newItem = new CartItem();
                    newItem.setCart(cart);
                    newItem.setProduct(product);
                    newItem.setPriceAtAdd(product.getPrice());
                    newItem.setQuantity(0);
                    return newItem;
                });

        String displayName = (dto.getAddedByDisplayName() != null &&
!dto.getAddedByDisplayName().isBlank())
                ? dto.getAddedByDisplayName() : "Гість";
        item.setAddedByDisplayName(displayName);
        item.setQuantity(item.getQuantity() + dto.getQuantity());
```

```
        cartItemRepository.save(item);

        Cart refreshed =
cartRepository.findById(cart.getId()).orElseThrow();
        return mapToDto(refreshed);
    }

    @Transactional
    public CartDto removeItemByToken(String token, Long productId,
String displayName) {
        Cart cart = cartRepository.findByShareToken(token)
                .orElseThrow(() -> new EntityNotFoundException("Cart
session not found: " + token));

        CartItem itemToRemove =
cartItemRepository.findByCartIdAndProductId(cart.getId(), productId)
                .orElseThrow(() -> new EntityNotFoundException("Item
not found in shared cart"));

        String productName = itemToRemove.getProduct().getName();

        cart.getItems().remove(itemToRemove);
        cartItemRepository.delete(itemToRemove);
        cartItemRepository.flush();

        CartDto result = mapToDto(cart);

        broadcastPartyEvent(token, CartEventMessage.Action.REMOVE,
displayName, productName, productId, null, result.getItems());
        return result;
    }

    @Transactional
    public CartDto updateItemQuantityByToken(String token, Long
productId, int quantity, String displayName) {
        Cart cart = cartRepository.findByShareToken(token)
                .orElseThrow(() -> new EntityNotFoundException("Cart
session not found: " + token));

        CartItem item =
cartItemRepository.findByCartIdAndProductId(cart.getId(), productId)
                .orElseThrow(() -> new EntityNotFoundException("Item
not found in shared cart"));

        String productName = item.getProduct().getName();

        if (quantity <= 0) {
            cartItemRepository.delete(item);
        } else {
            item.setQuantity(quantity);
            cartItemRepository.save(item);
        }
```

```
        Cart refreshed =
cartRepository.findById(cart.getId()).orElseThrow();
        CartDto result = mapToDto(refreshed);

        broadcastPartyEvent(token, CartEventMessage.Action.UPDATE,
displayName, productName, null, quantity > 0 ? quantity : null,
result.getItems());
        return result;
    }

    @Transactional
    public CartDto applyPromoByToken(String token, String code,
String displayName) {
        Cart cart = cartRepository.findByShareToken(token)
                .orElseThrow(() -> new EntityNotFoundException("Cart
session not found: " + token));

        if (cart.getPromoCode() != null &&
!cart.getPromoCode().isBlank()) {
            throw new RuntimeException("Промокод вже застосовано до
цього замовлення");
        }

        PromoCode promo =
promoCodeRepository.findByCodeAndIsActiveTrue(code.trim().toUpperCas
e())
                .orElseThrow(() -> new
EntityNotFoundException("Промокод не знайдено або він не
активний"));

        cart.setPromoCode(promo.getCode());
        cart.setDiscountPercentage(promo.getDiscountPercentage());
        cartRepository.save(cart);

        CartDto result = mapToDto(cart);

        CartEventMessage msg = new CartEventMessage();
        msg.setType(CartEventMessage.EventType.CART_UPDATED);
        msg.setAction(CartEventMessage.Action.PROMO_APPLIED);
        msg.setUserName(displayName != null ? displayName :
"Гість");
        msg.setPromoCode(promo.getCode());
        msg.setDiscountPercentage(promo.getDiscountPercentage());
        msg.setItems(result.getItems());
        messagingTemplate.convertAndSend("/topic/cart/" + token,
msg);

        return result;
    }

    @Transactional
    public CartDto cancelPromoByToken(String token, String
displayName) {
```

```
        Cart cart = cartRepository.findByShareToken(token)
                .orElseThrow(() -> new EntityNotFoundException("Cart
session not found: " + token));

        cart.setPromoCode(null);
        cart.setDiscountPercentage(null);
        cartRepository.save(cart);

        CartDto result = mapToDto(cart);

        CartEventMessage msg = new CartEventMessage();
        msg.setType(CartEventMessage.EventType.CART_UPDATED);
        msg.setAction(CartEventMessage.Action.PROMO_CANCELLED);
        msg.setUserName(displayName != null ? displayName :
"Гість");
        msg.setItems(result.getItems());
        messagingTemplate.convertAndSend("/topic/cart/" + token,
msg);

        return result;
    }

    public void broadcastLock(String token, String displayName) {
        CartEventMessage msg = new CartEventMessage();
        msg.setType(CartEventMessage.EventType.CART_UPDATED);
        msg.setAction(CartEventMessage.Action.LOCK_CART);
        msg.setUserName(displayName != null ? displayName :
"Гість");
        messagingTemplate.convertAndSend("/topic/cart/" + token,
msg);
    }

    @Transactional
    public OrderResponseDto placePartyOrder(String token,
PartyOrderDto dto) {
        Cart cart = cartRepository.findByShareToken(token)
                .orElseThrow(() -> new EntityNotFoundException("Cart
session not found: " + token));
        User owner = cart.getUser();

        boolean onlyMine =
"ONLY_MINE".equalsIgnoreCase(dto.getPaymentMode());

        List<CartItem> selectedItems = onlyMine
                ? cart.getItems().stream()
                        .filter(i -> dto.getUserName() != null &&
dto.getUserName().equals(i.getAddedByDisplayName()))
                        .toList()
                : new ArrayList<>(cart.getItems());

        if (selectedItems.isEmpty()) {
            throw new RuntimeException("Немає товарів для оплати");
        }
```

```
        Order order = new Order();
        order.setUser(owner);
        order.setStatus(OrderStatus.NEW);
        order.setItems(new ArrayList<>());

        BigDecimal total = BigDecimal.ZERO;
        for (CartItem ci : selectedItems) {
            Product product = ci.getProduct();
            if (product.getStockQuantity() < ci.getQuantity()) {
                throw new RuntimeException("Недостатньо залишків
товару: " + product.getName());
            }
            product.setStockQuantity(product.getStockQuantity() -
ci.getQuantity());
            productRepository.save(product);

            OrderItem oi = new OrderItem();
            oi.setOrder(order);
            oi.setProduct(product);
            oi.setQuantity(ci.getQuantity());
            oi.setPriceAtPurchase(ci.getPriceAtAdd());
            order.getItems().add(oi);

            total =
total.add(ci.getPriceAtAdd().multiply(BigDecimal.valueOf(ci.getQuant
ity())));
        }
        order.setTotalPrice(total);

        if (dto.getPromoCode() != null &&
!dto.getPromoCode().isBlank()) {
            String code = dto.getPromoCode().trim().toUpperCase();
            PromoCode promo =
promoCodeRepository.findByCodeAndIsActiveTrue(code)
                    .orElseThrow(() -> new
EntityNotFoundException("Промокод не знайдено або він не
активний"));
            BigDecimal multiplier = BigDecimal.valueOf(100 -
promo.getDiscountPercentage())
                    .divide(BigDecimal.valueOf(100));
            order.setPromoCode(code);
            order.setTotalPrice(total.multiply(multiplier));
        }

        orderRepository.save(order);

        cart.getItems().removeAll(selectedItems);
        cartItemRepository.deleteAll(selectedItems);
        cartItemRepository.flush();

        CartDto updatedCart = mapToDto(cart);
```

```
        if (onlyMine) {
            broadcastPartyEvent(token,
CartEventMessage.Action.UPDATE,
                    dto.getUserName(), null, null, null,
updatedCart.getItems());
        } else {
            CartEventMessage msg = new CartEventMessage();
            msg.setType(CartEventMessage.EventType.CART_UPDATED);
            msg.setAction(CartEventMessage.Action.ORDER_COMPLETED);
            msg.setUserName(dto.getUserName() != null ?
dto.getUserName() : "Гість");
            msg.setItems(Collections.emptyList());
            messagingTemplate.convertAndSend("/topic/cart/" + token,
msg);
        }

        OrderResponseDto response = new OrderResponseDto();
        response.setId(order.getId());
        response.setUserEmail(owner.getEmail());
        response.setStatus(order.getStatus());
        response.setTotalPrice(order.getTotalPrice());
        response.setCreatedAt(order.getCreatedAt());
        return response;
    }

    private void broadcastPartyEvent(String token,
CartEventMessage.Action action,
                                     String displayName, String
productName,
                                     Long productId, Integer
quantity, List<CartItemDto> items) {
        CartEventMessage msg = new CartEventMessage();
        msg.setType(CartEventMessage.EventType.CART_UPDATED);
        msg.setAction(action);
        msg.setUserName(displayName != null ? displayName :
"Гість");
        msg.setProductName(productName);
        msg.setProductId(productId);
        msg.setQuantity(quantity);
        msg.setItems(items);
        messagingTemplate.convertAndSend("/topic/cart/" + token,
msg);
    }

    private Cart getOrCreateCartEntity(User user) {
        return
cartRepository.findByUserId(user.getId()).orElseGet(() -> {
            Cart cart = new Cart();
            cart.setUser(user);
            return cartRepository.save(cart);
        });
    }
```

```
    private User getUserByEmail(String email) {
        return userRepository.findByEmail(email)
                .orElseThrow(() -> new EntityNotFoundException("User
not found: " + email));
    }

    public CartDto mapToDto(Cart cart) {
        CartDto dto = new CartDto();
        dto.setCartId(cart.getId());
        dto.setOwnerId(cart.getUser().getId());
        dto.setOwnerEmail(cart.getUser().getEmail());
        dto.setShareToken(cart.getShareToken());
        dto.setPromoCode(cart.getPromoCode());
        dto.setDiscountPercentage(cart.getDiscountPercentage());

        List<CartItemDto> itemDtos =
cart.getItems().stream().map(item -> {
            CartItemDto itemDto = new CartItemDto();
            itemDto.setProductId(item.getProduct().getId());
            itemDto.setProductName(item.getProduct().getName());

itemDto.setProductImageUrl(item.getProduct().getImageUrl());
            itemDto.setQuantity(item.getQuantity());
            itemDto.setPriceAtAdd(item.getPriceAtAdd());

itemDto.setTotalPrice(item.getPriceAtAdd().multiply(BigDecimal.value
Of(item.getQuantity())));

itemDto.setAddedByDisplayName(item.getAddedByDisplayName());
            return itemDto;
        }).toList();

        dto.setItems(itemDtos);
        dto.setTotalPrice(itemDtos.stream()
                .map(CartItemDto::getTotalPrice)
                .reduce(BigDecimal.ZERO, BigDecimal::add));

        return dto;
    }
}
```

Додаток И. Лістинг файлу «CartContext.jsx»

```
import { createContext, useContext, useEffect, useMemo, useState }
from 'react'

export const CartContext = createContext(null)

const getInitialCart = () => {
  try {
    const savedCart = localStorage.getItem('cart')
    return savedCart ? JSON.parse(savedCart) : []
  } catch (error) {
    console.error('Error parsing cart from localStorage:', error)
    return []
  }
}

export function CartProvider({ children }) {
  const [cartItems, setCartItems] = useState(getInitialCart)

  useEffect(() => {
    try {
      localStorage.setItem('cart', JSON.stringify(cartItems))
    } catch (error) {
      console.error('Error saving cart to localStorage:', error)
    }
  }, [cartItems])

  const addToCart = (product) => {
    setCartItems(prevItems => {
      const existingItem = prevItems.find(item => item.id ===
product.id)

      if (existingItem) {
        return prevItems.map(item =>
          item.id === product.id
            ? { ...item, quantity: item.quantity + 1 }
            : item
        )
      } else {
        return [...prevItems, { ...product, quantity: 1 }]
      }
    })
  }

  const removeFromCart = (productId) => {
    setCartItems(prevItems => prevItems.filter(item => item.id !==
productId))
  }

  const updateQuantity = (productId, amount) => {
    setCartItems(prevItems => {
```

```
      return prevItems
        .map(item => {
          if (item.id === productId) {
            const newQuantity = item.quantity + amount
            return newQuantity > 0 ? { ...item, quantity:
newQuantity } : null
          }
          return item
        })
        .filter(Boolean)
    })
  }

  const clearCart = () => {
    setCartItems([])
  }

  const getCartTotal = () => {
    return cartItems.reduce((total, item) => total + (item.price *
item.quantity), 0)
  }

  const getCartItemsCount = () => {
    return cartItems.reduce((total, item) => total + item.quantity,
0)
  }

  const value = useMemo(
    () => ({
      cartItems,
      addToCart,
      removeFromCart,
      updateQuantity,
      clearCart,
      getCartTotal,
      getCartItemsCount,
    }),
    [cartItems],
  )

  return <CartContext.Provider
value={value}>{children}</CartContext.Provider>
}

export function useCart() {
  const context = useContext(CartContext)

  if (!context) {
    throw new Error('useCart must be used within CartProvider')
  }

  return context}
```

Додаток Й. Лістинг файлу «ProductDetailsPage.jsx»

```
import { useEffect, useState } from "react";
import { useParams, Link } from "react-router-dom";
import { Star, Network } from "lucide-react";
import api from "../api/axios";
import { useCart } from "../context/CartContext";
import { useAuth } from "../context/AuthContext";
import { useSharedCart } from "../hooks/useSharedCart";
import IngredientGraphModal from
"../components/IngredientGraphModal";
import IngredientDetailsModal from
"../components/IngredientDetailsModal";
import { getIngredientCategoryColor } from
"../constants/ingredientCategoryColors";

function Spinner() {
  return (
    <div className="flex flex-col items-center justify-center min-h-
[50vh]">
      <svg
        className="animate-spin h-10 w-10 text-rose-300 mb-2"
        xmlns="http://www.w3.org/2000/svg"
        fill="none"
        viewBox="0 0 24 24"
      >
        <circle
          className="opacity-30"
          cx="12"
          cy="12"
          r="10"
          stroke="currentColor"
          strokeWidth="4"
        ></circle>
        <path
          className="opacity-70"
          fill="currentColor"
          d="M4 12a8 8 0 018-8v4a4 4 0 00-4 4H4z"
        ></path>
      </svg>
      <span className="text-rose-400 font-
medium">Завантаження...</span>
    </div>
  );
}

function ProductDetailsPage() {
  const { productId } = useParams();
  const [product, setProduct] = useState(null);
  const [isLoading, setIsLoading] = useState(true);
  const [isAdded, setIsAdded] = useState(false);
  const { addToCart } = useCart();
```

```
  const { user } = useAuth();
  const { shareToken, addToSharedCart } = useSharedCart();

  const [ingredients, setIngredients] = useState([]);
  const [ingredientsLoading, setIngredientsLoading] =
useState(false);
  const [selectedIngredient, setSelectedIngredient] =
useState(null);
  const [isIngredientModalOpen, setIsIngredientModalOpen] =
useState(false);
  const [isGraphOpen, setIsGraphOpen] = useState(false);

  const [reviews, setReviews] = useState([]);
  const [reviewsLoading, setReviewsLoading] = useState(false);
  const [reviewRating, setReviewRating] = useState(5);
  const [reviewComment, setReviewComment] = useState('');
  const [reviewMessage, setReviewMessage] = useState('');
  const [reviewError, setReviewError] = useState('');
  const [hoverRating, setHoverRating] = useState(0);

  const handleAddToCart = async () => {
    if (shareToken) {
      await addToSharedCart(product);
    } else {
      addToCart(product);
    }
    setIsAdded(true);
    setTimeout(() => setIsAdded(false), 1500);
  };

  useEffect(() => {
    setIsLoading(true);
    setProduct(null);
    const fetchProduct = async () => {
      try {
        const response = await api.get(`/products/${productId}`);
        setProduct(response.data);
      } catch (e) {
        setProduct(null);
      } finally {
        setIsLoading(false);
      }
    };
    fetchProduct();
  }, [productId]);

  useEffect(() => {
    const fetchReviews = async () => {
      try {
        setReviewsLoading(true);
        const res = await api.get(`/products/${productId}/reviews`);
        setReviews(res.data || []);
      } catch (e) {
```

```
        setReviews([]);
      } finally {
        setReviewsLoading(false);
      }
    };

    fetchReviews();
  }, [productId]);

  useEffect(() => {
    const fetchIngredients = async () => {
      try {
        setIngredientsLoading(true);
        const res = await
api.get(`/products/${productId}/ingredients`);
        setIngredients(res.data || []);
      } catch (e) {
        setIngredients([]);
      } finally {
        setIngredientsLoading(false);
      }
    };

    fetchIngredients();
  }, [productId]);

  const handleSubmitReview = async (e) => {
    e.preventDefault();
    setReviewMessage('');
    setReviewError('');

    try {
      await api.post('/reviews', {
        productId: Number(productId),
        rating: Number(reviewRating),
        comment: reviewComment,
      });

      setReviewMessage('Дякуємо! Ваш відгук додано.');
      setReviewComment('');
      setReviewRating(5);

      const res = await api.get(`/products/${productId}/reviews`);
      setReviews(res.data || []);

      const prodRes = await api.get(`/products/${productId}`);
      setProduct(prodRes.data);
    } catch (err) {
      setReviewError(err.response?.data?.message || 'Помилка
додавання відгуку');
    }
  };
```

```
  return (
    <main className="min-h-screen bg-rose-50 px-6 py-16 text-zinc-
900">
      <section className="mx-auto max-w-7xl rounded-2xl border
border-rose-100 bg-white p-8 shadow-sm">
        <h1 className="mb-8 text-3xl font-bold"></h1>

        {isLoading ? (
          <Spinner />
        ) : !product ? (
          <div className="text-center text-lg text-zinc-500 my-20">
            Товар не знайдено
          </div>
        ) : (
          <div className="grid grid-cols-1 md:grid-cols-2 gap-12">
            <div className="flex items-center justify-center bg-
white rounded-2xl shadow h-[500px] p-3">
              {product.imageUrl?.trim() ? (
                <img
                  src={product.imageUrl}
                  alt={product.name}
                  className="h-full w-full object-contain mix-blend-
multiply rounded-2xl"
                />
              ) : (
                <div className="flex h-full w-full items-center
justify-center rounded-2xl bg-pink-100">
                  <span className="text-lg font-medium text-rose-
400">
                    Немає фото
                  </span>
                </div>
              )}
            </div>

            <div className="flex flex-col justify-start">
              <Link
                to="/"
                className="
                  inline-block mb-5 text-rose-400 hover:text-rose-
600
                  transition-colors font-medium text-md
                "
              >
                ← Назад до каталогу
              </Link>
              <h2 className="text-3xl font-bold text-gray-800 mb-
2">{product.name}</h2>
              {Number(product.reviewCount) > 0 && (
                <div className="text-sm text-zinc-600 mb-2">
                  ⭐ {Number(product.averageRating).toFixed(1)}
({product.reviewCount} відгуків)
```

```
              </div>
            )}
            <div className="text-gray-600 my-
4">{product.description || "—"}</div>
            <div className="text-2xl font-bold text-rose-500 mb-
6">
              {product.price} ₴
            </div>
            <button
              onClick={handleAddToCart}
              className={`
                w-full py-3 rounded-xl
                mt-6 text-lg font-semibold
                shadow transition-all duration-150
                ${isAdded
                  ? 'bg-green-500 hover:bg-green-600 text-white'
                  : 'bg-rose-500 hover:bg-rose-600 text-white'
                }
              `}
            >
              {isAdded ? '✓ Додано!' : 'Додати в кошик'}
            </button>
          </div>
        </div>
      )}
    </section>

    {!isLoading && product && (
      <section className="mx-auto max-w-7xl mt-8 rounded-2xl
border border-rose-100 bg-white p-8 shadow-sm">
        <div className="flex flex-col md:flex-row md:items-center
md:justify-between gap-4 mb-6">
          <h3 className="text-2xl font-bold text-gray-800">Склад
продукту</h3>
          <button
            onClick={() => setIsGraphOpen(true)}
            className="inline-flex items-center justify-center
gap-2 rounded-xl bg-gradient-to-r from-rose-500 to-pink-500 px-4 py-
2.5 text-sm font-semibold text-white shadow hover:opacity-95
transition"
          >
            <Network className="w-4 h-4" />
            Інтерактивна карта інгредієнтів
          </button>
        </div>

        {ingredientsLoading ? (
          <div className="text-zinc-500">Завантаження
складу...</div>
        ) : ingredients.length === 0 ? (
          <div className="text-zinc-500">Інгредієнти не
вказані.</div>
```

```
          ) : (
            <div className="flex flex-wrap gap-2">
              {ingredients.map((ing) => (
                <button
                  key={ing.id}
                  type="button"
                  onClick={() => {
                    setSelectedIngredient(ing)
                    setIsIngredientModalOpen(true)
                  }}
                  style={{ backgroundColor:
getIngredientCategoryColor(ing.category) }}
                  className="rounded-full px-3 py-1.5 text-sm font-
semibold text-gray-800 shadow-sm hover:opacity-95 transition"
                >
                  {ing.name}
                </button>
              ))}
            </div>
          )}
        </section>
      )}

      {!isLoading && product && (
        <section className="mx-auto max-w-7xl mt-8 rounded-2xl
border border-rose-100 bg-white p-8 shadow-lg">
          <h3 className="text-2xl font-bold text-gray-800 mb-
6">Відгуки клієнтів</h3>

          {reviewsLoading ? (
            <div className="text-zinc-500">Завантаження
відгуків...</div>
          ) : reviews.length === 0 ? (
            <div className="text-zinc-500">Поки що немає
відгуків.</div>
          ) : (
            <div className="space-y-4">
              {reviews.map((r) => (
                <div key={r.id} className="rounded-xl border border-
rose-100 p-4">
                  <div className="flex items-center justify-between
mb-1">
                    <div className="font-semibold text-gray-
800">{r.authorName || 'Користувач'}</div>
                    <div className="text-sm text-zinc-500">
                      {r.createdAt ? new
Date(r.createdAt).toLocaleDateString('uk-UA') : ''}
                    </div>
                  </div>
                  <div className="text-sm text-zinc-600 mb-
2">{'⭐'.repeat(Number(r.rating || 0))}</div>
                  <div className="text-gray-700 whitespace-pre-
```

```
wrap">{r.comment || ''}</div>
                </div>
              ))}
            </div>
          )}

          <div className="mt-8">
            {user ? (
              <div className="rounded-xl border border-rose-100 p-5
bg-rose-50">
                <h4 className="text-lg font-semibold text-gray-800
mb-4">Залишити відгук</h4>

                {reviewMessage && (
                  <div className="mb-4 rounded-lg border border-
green-200 bg-green-50 p-3 text-green-800">
                    {reviewMessage}
                  </div>
                )}
                {reviewError && (
                  <div className="mb-4 rounded-lg border border-red-
200 bg-red-50 p-3 text-red-800">
                    {reviewError}
                  </div>
                )}

                <form onSubmit={handleSubmitReview}
className="space-y-4">
                  <div>
                    <label className="block text-sm font-medium
text-gray-700 mb-2">Рейтинг</label>
                    <div className="flex gap-1">
                      {[1, 2, 3, 4, 5].map((starValue) => (
                        <button
                          key={starValue}
                          type="button"
                          onClick={() => setReviewRating(starValue)}
                          onMouseEnter={() =>
setHoverRating(starValue)}
                          onMouseLeave={() => setHoverRating(0)}
                          className="cursor-pointer transition-
colors"
                        >
                          <Star
                            className={`w-6 h-6 ${
                              starValue <= (hoverRating ||
reviewRating)
                                ? 'text-yellow-400 fill-yellow-400'
                                : 'text-gray-300 fill-transparent'
                            }`}
                          />
                        </button>
                      ))}
```

```
                  </div>
                </div>

                <div>
                  <label className="block text-sm font-medium
text-gray-700 mb-2">Коментар</label>
                  <textarea
                    value={reviewComment}
                    onChange={(e) =>
setReviewComment(e.target.value)}
                    rows={4}
                    className="w-full rounded-lg border border-
rose-100 bg-white px-3 py-2 text-sm outline-none transition
focus:border-rose-300"
                    placeholder="Поділіться враженнями про
товар..."
                  />
                </div>

                <button
                  type="submit"
                  className="rounded-lg bg-rose-500 px-4 py-2.5
text-sm font-medium text-white transition hover:bg-rose-600"
                >
                  Залишити відгук
                </button>
              </form>
            </div>
          ) : (
            <div className="rounded-xl border border-rose-100 p-5
bg-rose-50 text-zinc-700">
              Увійдіть або зареєструйтесь, щоб залишити відгук.
            </div>
          )}
        </div>
      </section>
    )}

    <IngredientDetailsModal
      ingredient={selectedIngredient}
      isOpen={isIngredientModalOpen}
      onClose={() => setIsIngredientModalOpen(false)}
    />

    <IngredientGraphModal
      productId={productId}
      isOpen={isGraphOpen}
      ingredients={ingredients}
      onClose={() => setIsGraphOpen(false)}
    />
  </main>
); } export default ProductDetailsPage;
```

2026.КВР.122.421.09.00.00БД

users

PK	id bigint
	email varchar(255)
	password varchar(255)
	first_name varchar(100)
	last_name varchar(100)
	role varchar(50)
	phone_number varchar(20)
	address text
	created_at timestamp

orders

PK	id bigint
	user_id bigint
	status varchar(50)
	total_price numeric(38,2)
	promo_code varchar(255)
	created_at timestamp

order_items

PK	id bigint
	order_id bigint
	product_id bigint
	quantity integer
	price_at_purchase numeric(38,2)

products

PK	id bigint
	name varchar(255)
	description text
	price numeric(38,2)
	stock_quantity integer
	category_id bigint
	brand_id bigint
	created_at timestamp
	image_url varchar(255)

product_ingredients

	product_id bigint
	ingredient_id bigint

categories

PK	id bigint
	name varchar(255)
	description text

brands

PK	id bigint
	name varchar(255)
	country varchar(100)

carts

PK	id bigint
	user_id bigint
	share_token varchar(64)
	discount_percentage integer
	promo_code varchar(255)
	created_at timestamp

cart_items

PK	id bigint
	cart_id bigint
	product_id bigint
	quantity integer
	price_at_add numeric(38,2)
	added_by_display_name varchar(100)

promo_codes

PK	id bigint
	code varchar(255)
	discount_percentage integer
	is_active boolean

reviews

PK	id bigint
	user_id bigint
	product_id bigint
	rating integer
	comment text
	created_at timestamp
	created_at timestamp

ingredient_interactions

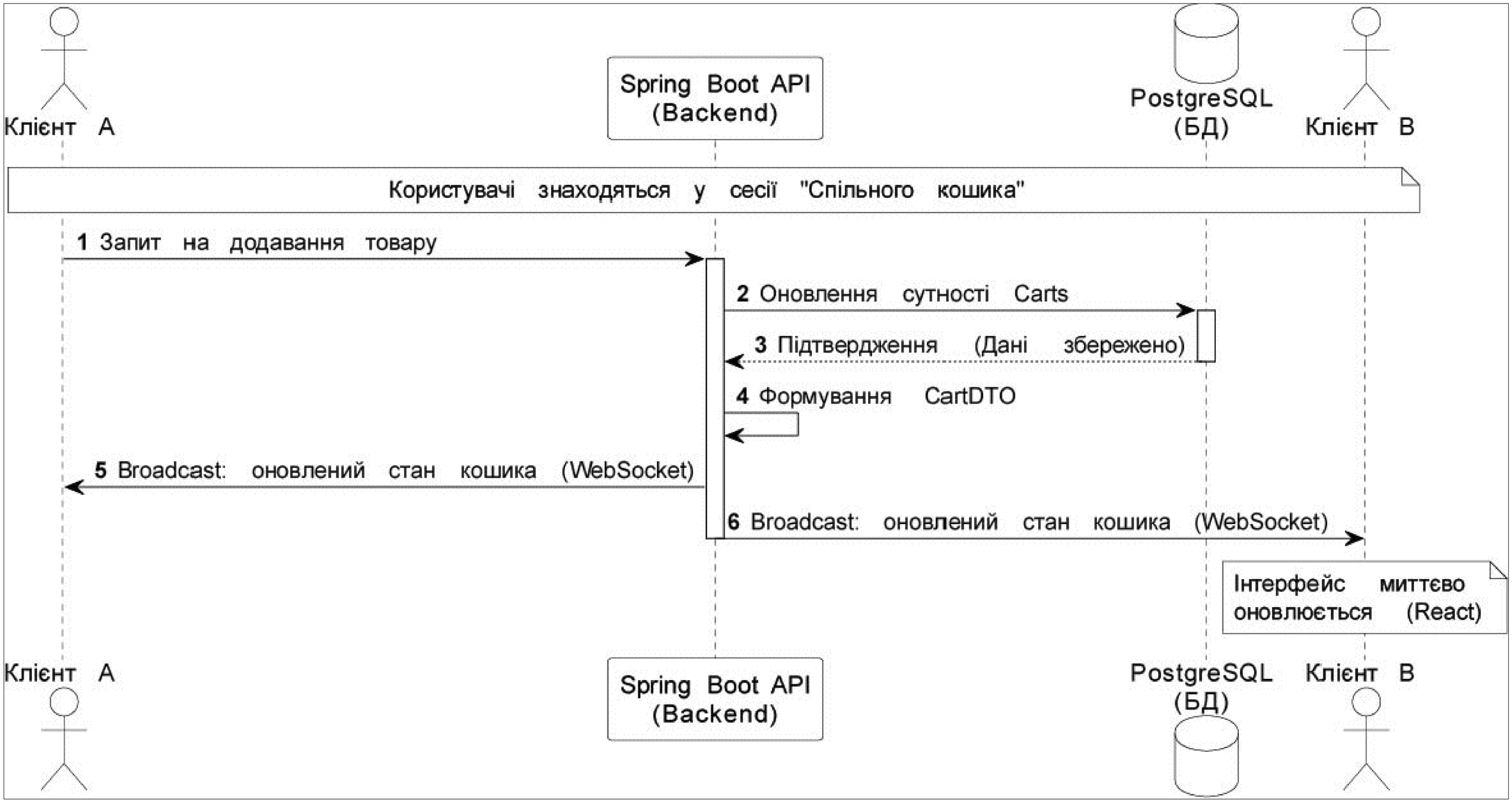
PK	id bigint
	ingredient1_id bigint
	ingredient2_id bigint
	type varchar(255)
	description text

ingredients

PK	id bigint
	name varchar(255)
	category varchar(255)
	comedogenic_rating integer
	safety_rating integer
	description text

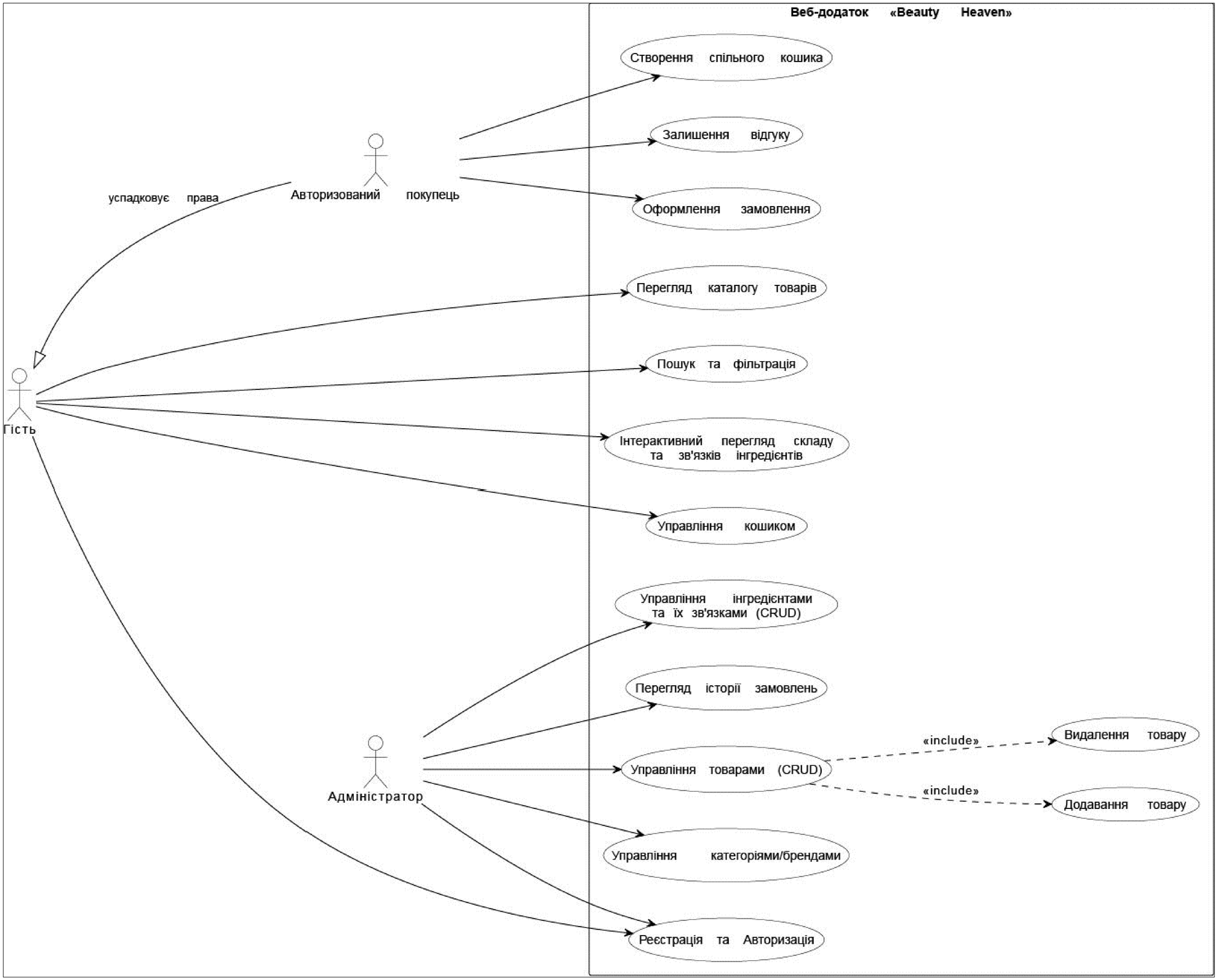
Зм.	Арк.	№ докум.	Підпис	Дата	2026.КВР.122.421.09.00.00БД		
Розроб.		Іваход С.О.			Розробка інтернет-магазину косметики "BeautyHeaven"	Літ.	Маса	Масштаб
Перевір.		Сербін В.С.			ER-діаграма бази даних			
Т.контр.						Аркуш	Аркушів 1	
Рецензент						ВСП ТФК ТНТУ КН-421		
Н.контр.		Приймак В.А.				м. Тернопіль		
Затв.								

Копіював Формат А1



					2026.КВР.122.4.21.09.00.00.ДП	
Зм.	Арк.	№ докум.	Підпис	Дата	Розробка інтернет-магазину косметики "BeautyHeaven" Діаграма послідовності (Sequence)	Літ. / Маса / Масштаб
Розроб.		Івахов С.О.				
Перевір.		Сербін В.С.				Аркуш / Аркушів 1
Т.контр.						
Рецензент						ВСП ТФК ТНТУ КН-4.21 м. Тернопіль
Н.контр.		Приймак В.А.				
Затв.						

Копіював Формат А1



						2026.КВР.122.421.09.00.00.ДВ	
Зм.	Арк.	№ докум.	Підпис	Дата	Розробка інтернет-магазину косметики "BeautyHeaven"	Літ.	Маса	Масштаб
Розроб.		Івахов С.О.			Діаграма прецедентів (Use case)			
Перевір.		Сердін В.С.				Аркуш	Аркушів 1	
Т.контр.						ВСП ТФК ТНТУ КН-421 м. Тернопіль		
Рецензент								
Н.контр.		Приймак В.А.						
Затв.								

Копіював Формат А1

Таблиця техніко-економічних показників

№ п/п	Показник	Одиниці вимірювання	Значення
1	Мова програмування	-	Java, JavaScript
2	Технології	-	Spring Boot, React, Tailwind CSS, Vite
3	Менеджер пакетів	-	Maven, npm
4	Система керування БД	-	PostgreSQL
5	Архітектура програмного забезпечення	-	клієнт-серверна
6	Загальний обсяг вихідного коду	МБ	175
7	Трудомісткість розробки	год	158
8	Чиста теперішня вартість	грн.	35 494
9	Собівартість	грн.	90 328
10	Очікуваний прибуток	грн.	50 584
11	Термін окупності	рік	2,1