Вищий навчальний заклад

«Університет економіки та права «КРОК»
Фаховий коледж

Циклова комісія з інформаційних технологій

Кваліфікаційна робота фахового молодшого бакалавра

на тему Створення інтернет-магазину з продажу товарів доглядової косметики

Виконав_____
(Підпис)
__Зволинський Владислав Сергійович____
(прізвище, ім'я, по батькові)

_____

Науковий керівник_____
_____Кириченко Віктор Вікторович____
(прізвище, ім'я, по батькові)

_____

_____
(Резолюція «До захисту»)

Попередній захист:

_____
(Висновок: "До захисту в екзаменаційній комісії ")

Голова циклової комісії _____ _____ _____
(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2025 року

ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

УНІВЕРСИТЕТ ЕКОНОМІКИ ТА ПРАВА «КРОК»

Фаховий коледж

Циклова комісія з інформаційних технологій

Спеціальність 121 інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Голова циклової комісії _____ Леонід УВАРОВ
(підпис)

«____» _____ 2025 року

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Здобувач освіти Зволинський Владислав Сергійович

1. Тема роботи Створення інтернет-магазину з продажу товарів доглядової косметики затверджена наказом по університету від «___» _____ 202__ р. № _____
2. Термін здачі закінченої роботи «30» травня 2025 року
3. Вихідні дані до роботи:
    1) Функціональні вимоги до інтернет-магазину: перегляд товарів, додавання до кошика, реєстрація, оформлення замовлення, інтеграція з платіжними сервісами.
    2) Використовувані технології: HTML, CSS, JavaScript, Bootstrap (фронтенд); Node.js, Express.js, JSON, bcrypt, JWT (бекенд).
    3) Цільова аудиторія – споживачі доглядової косметики в Україні.
    4) Аналіз популярних рішень: Shopify, MakeUp.ua, Eva.ua.
    5) Забезпечення адаптивності інтерфейсу, зручності користування та базової безпеки.
4. Зміст пояснювальної записки
    1) **Розділ 1. Теоретична частина**
    Аналіз сучасних інтернет-магазинів у сфері доглядової косметики. Порівняння платформ. Обґрунтування вибору технологій.
    2) **Розділ 2. Проєктування та розробка**
    Архітектура системи, реалізація інтерфейсу користувача, клієнтської та серверної частини, опис алгоритмів реєстрації, входу, додавання товарів, обробки замовлень.
    3) **Розділ 3. Експериментальна частина**
    Тестування функціональності на різних пристроях, виправлення помилок, інструкції користувача, аналіз ефективності, рекомендації щодо подальшого розвитку системи.

5. Перелік графічного матеріалу:

   Скріншоти існуючих рішень

   Скріншоти інтерфейсу продукту

   Схеми і таблиці щодо візуалізації аналізу

   Блок-схеми алгоритмів

   Діаграми щодо проєктування продукту (наприклад, потоків даних, переходів станів, сутність-зв'язок, UML, бази даних тощо)

Дата видачі завдання «12» лютого 2025 року

Науковий керівник _____ Кириченко В. В.
(підпис) (прізвище, ім'я, по батькові)

Завдання прийняв до виконання _____ Зволинський В. С
(підпис) (прізвище, ім'я, по батькові)

РЕФЕРАТ

Пояснювальна записка: 46 сторінок, 6 рисунків, 4 таблиць, 5 додатків, 2 джерел.
Об'єкт дослідження – створення інтернет-магазину з продажу доглядової косметики.
Мета роботи – розробка інтерфейсу що забезпечує управління товарами, обробку замовлень, реєстрацію користувачів та інтеграцію з платіжними сервісами.
У кваліфікаційній роботі проаналізовано сучасні підходи до реалізації онлайн-торгівлі, вивчено популярні платформи та технології електронної комерції. На основі отриманих даних створено прототип інтернет-магазину з використанням мови програмування JavaScript, HTML, SCC. Інтерфейс реалізовано з використанням HTML, CSS та бібліотеки Bootstrap.
Система підтримує функціонал перегляду товарів, реєстрації користувачів, оформлення замовлень. У роботі описано архітектуру веб-додатку, реалізовані модулі, інтерфейс користувача. Систему протестовано на різних пристроях, підготовлено технічну документацію та надано рекомендації щодо її подальшого розвитку.
Результати розробки можуть бути використані для створення або вдосконалення веб-магазинів у сфері продажу доглядової косметики.
Ключові слова: інтернет-магазин, доглядова косметика, JavaScript,, Bootstrap, управління товарами, замовлення.

ABSTRACT

Explanatory note: 46 pages, 6 figures, 4 tables, 5 appendices, 2 sources.
The object of the study is the creation of an online store for the sale of care cosmetics.
The purpose of the work is to develop an interface that provides product management, order processing, user registration and integration with payment services.
The qualification work analyzes modern approaches to the implementation of online trading, studies popular e-commerce platforms and technologies. Based on the data obtained, a prototype of an online store was created using the JavaScript, HTML, SCC programming languages. The interface is implemented using HTML, CSS and the Bootstrap library.
The system supports the functionality of viewing products, user registration, and placing orders. The work describes the architecture of the web application, implemented modules, and user interface. The system was tested on various devices, technical documentation was prepared, and recommendations for its further development were provided.
The results of the development can be used to create or improve web stores in the field of selling skincare cosmetics.
Keywords: online store, skincare cosmetics, JavaScript, Bootstrap, product management, orders.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API — Application Programming Interface (інтерфейс програмування застосунків)

CSS — Cascading Style Sheets (каскадні таблиці стилів)

GUI — Graphical User Interface (графічний інтерфейс користувача)

UI — User Interface (користувацький інтерфейс)

UX — User Experience (користувацький досвід)

HTML — HyperText Markup Language (мова розмітки гіпертексту)

JS — JavaScript (мова програмування)

Bootstrap — Безкоштоний фреймворк за набором попередньо зроблених елементів та компонентів

Express – Основний веб-фрейморк для Node.js

Bcrypt – Бібліотека для хешування паролів

ВСТУП

Актуальність завдання. Сучасний ринок доглядової косметики динамічно розвивається, демонструючи стійке зростання попиту серед споживачів [3]. Збільшення кількості брендів та продуктів зумовлює високу конкуренцію серед компаній, які пропонують засоби для догляду за обличчям, тілом та волоссям [4]. В умовах цифрової трансформації бізнесу, зокрема інтернет-магазини, стають ключовим інструментом для забезпечення зручного, швидкого та ефективного процесу онлайн-покупок [5].

Мета роботи. Розробити систему продажу доглядової косметики у вигляді інтернет-магазину, яка забезпечує інтуїтивно зрозумілий інтерфейс для користувачів, можливість оформлення замовлень онлайн та реєстрації [6].

Завдання роботи.

- Проаналізувати існуючі рішення та кращі практики у сфері онлайн-продажу доглядової косметики [7].
- Обрати оптимальні технології та платформу для розробки інтернет-магазину з урахуванням вимог функціональності [8].
- Спроєктувати архітектуру та інтерфейс користувача [9];
- Провести тестування та виправлення помилок [10];
- Підготувати інструкції для користувачів і технічного персоналу [11].

Об’єкт дослідження. Інтернет-магазини з продажу товарів доглядової косметики.

Предмет дослідження. Методи та технології розробки інтерактивних веб-додатків для реалізації функцій онлайн-продажу доглядової косметики з урахуванням вимог безпеки, зручності користування та ефективної взаємодії з користувачами [12].

Методи дослідження. Аналіз наукової літератури та існуючих рішень, порівняльний аналіз функціональності, моделювання системи, прототипування інтерфейсу, програмна реалізація, тестування та налагодження [13].

Практичне значення одержаних результатів. Розроблена система може бути використана для автоматизації продажів у невеликих та середніх інтернет-магазинах, що спеціалізуються на доглядовій косметиці. Її впровадження сприятиме підвищенню ефективності комерційної діяльності, поліпшенню користувацького досвіду, забезпеченню швидкої обробки замовлень та зменшенню впливу людського фактору в процесах продажу [14].

Структура роботи. Дипломна робота складається зі вступу, трьох основних розділів, висновків, списку використаних джерел та додатків. У першому розділі наведено теоретичний аналіз існуючих рішень і вибір технологій. Другий розділ присвячено проєктуванню та розробці системи, а третій — експериментальній перевірці, тестуванню та оформленню документації [15].

РОЗДІЛ 1. ТЕОРЕТИЧНА ЧАСТИНА. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ВИБІР ТЕХНОЛОГІЙ

1.1. Огляд сучасних інтернет-магазинів

Сучасні інтернет-магазини є невід'ємною частиною електронної комерції та широко використовуються для оптимізації процесів реалізації товарів і послуг [3]. У контексті ринку доглядової косметики ці системи дозволяють компаніям мінімізувати витрати, прискорити обробку замовлень, підвищити якість обслуговування клієнтів та ефективніше взаємодіяти з покупцями [4].

Інтернет-магазини доглядової косметики зазвичай містять такі ключові компоненти:

- каталог товарів із можливістю фільтрації й сортування;
- модуль оформлення замовлень;
- інтеграцію з платіжними сервісами та службами доставки [5];
- базу даних клієнтів і їхніх замовлень (CRM);
- аналітичний модуль для відстеження продажів;
- інструменти маркетингу: знижки, акції, система рекомендацій [6];
- технічну підтримку користувачів через чат або форму зворотного зв'язку.

У процесі дослідження проаналізовано популярні рішення, що активно використовуються в Україні та за кордоном у сфері продажу косметичних товарів:

- MakeUp.ua — один з найбільших інтернет-магазинів косметики в Україні. Пропонує зручний каталог, фільтрацію за категоріями, брендовану продукцію, відгуки, рекомендації;
- Eva.ua — мультибрендовий магазин з доглядовою косметикою, що включає програму лояльності, мобільний додаток і швидку доставку;
- Shopify — глобальна платформа для створення інтернет-магазинів, що підтримує повну автоматизацію процесу продажу,

включаючи інвентаризацію, виставлення рахунків, доставку та аналітику [7].

Також варто згадати CRM-системи, такі як Bitrix24, які активно використовуються в поєднанні з системами продажу. Вони дозволяють керувати базою клієнтів, відстежувати історію покупок, автоматизувати розсилки та обслуговування [8].

Ключовими перевагами сучасних систем продажу є:

- оперативність: обробка великої кількості замовлень без затримок;
- індивідуальний підхід: персоналізовані рекомендації на основі поведінки користувача [9];
- цілодобова доступність: доступ до товарів у будь-який час;
- інтеграція з іншими сервісами: взаємодія з логістичними, фінансовими, маркетинговими системами [10].

1.1.1. Характеристики популярних рішень

Система	**Тип платформи**	**Простота використання**	**Масштабованість**	**Підтримка мов/валют**	**Гнучкість налаштувань**	**Вартість**
MakeUp.ua	Власна	Висока	Висока	UA	Низька	Безкоштовна для користувача
Comfy	Власна	Висока	Середня	UA	Середня	Безкоштовна для користувача
Shopify	SaaS	Дуже висока	Висока	Так	Середня	$29+/міс.

Аналіз популярних платформ дозволяє виділити функціональні елементи, які доцільно реалізовувати у власному інтернет-магазині доглядової косметики: інтуїтивний інтерфейс, швидке оформлення замовлень, інтеграція з платіжними системами, система фільтрів, блок рекомендацій та підтримка користувачів [11].

1.1.2. Порівняльний аналіз функціональних можливостей

Для вибору оптимальної платформи онлайн-продажу доглядової косметики проведено порівняльний аналіз основних функціональних можливостей популярних рішень, що застосовуються у сфері електронної комерції. Аналіз базується на ключових параметрах, які впливають на продуктивність системи, рівень зручності для користувачів, гнучкість налаштування та масштабованість. До основних функцій, що аналізувалися, належать:

- **Обробка замовлень:** функціонал оформлення замовлення, вибору способу оплати і доставки.
- **Інтерфейс користувача:** зручність навігації, адаптивність під різні пристрої (ПК, мобільні телефони, планшети).
- **Аналітика та звітність:** збір статистики продажів, поведінки користувачів, формування звітів.
- **Безпека:** захист персональних даних, шифрування, підтримка SSL, багаторівнева автентифікація.
- **Підтримка мультимовності та мультивалюти:** важливо для роботи на міжнародних ринках.

Нижче наведено узагальнену таблицю порівняння функціональності основних платформ:

Функція	**MakeUp.ua**	**Eva.ua**	**Shopify**
Каталог товарів	+	+	+
Обробка замовлень	+	+	+
Інтерфейс користувача	+	+	+
Інтеграція з платіжними системами	+	+	+
Управління клієнтами (CRM)	+	+	+

Аналітика та звітність	+	+	+
Безпека	+	+	+
Масштабованість	Середня	Середня	Висока
Підтримка мультимовності та мультивалюти	Обмежена	Обмежена	Так

Пояснення позначок:

"+" — повна підтримка;

"±" — обмежена або базова підтримка;

"-" — відсутність підтримки.

Результати аналізу свідчать про те, що системи Shopify і Magento мають найбільш повний набір функцій для реалізації високоякісного онлайн-магазину доглядової косметики, проте для невеликих і середніх підприємств найкращим компромісом між функціональністю та простотою впровадження є використання OpenCart. Ця система забезпечує достатній рівень кастомізації, підтримує популярні способи оплати, має простий інтерфейс адміністратора та велику кількість безкоштовних модулів, що ідеально підходить для цілей розробки індивідуального рішення в межах дипломного проєкту [13].

Системи **MakeUp.ua** та **Eva.ua** орієнтовані більше на ринок України і мають специфічний функціонал, пристосований до локальних особливостей, але їх використання обмежується можливостями платформи.

Отже, при розробці власної автоматизованої системи продажу мобільних телефонів доцільно поєднати кращі функціональні рішення цих платформ з урахуванням специфіки цільової аудиторії та технічних можливостей команди розробників.

1.2. Аналіз кращих практик у сфері онлайн-продажів мобільних телефонів

Аналіз кращих практик у сфері онлайн-продажів косметичних товарів

Ринок косметики є одним із найбільш динамічних і конкурентних секторів електронної комерції. Для успішного функціонування автоматизованих систем продажу в цій сфері важливо враховувати кращі практики, які застосовують провідні онлайн-магазини та платформи.

Зручність і швидкість оформлення замовлення. Сучасні покупці очікують інтуїтивно зрозумілий інтерфейс, який дозволяє швидко знаходити потрібні товари, порівнювати їх характеристики, додавати обрані продукти у кошик та оформлювати замовлення за кілька кліків. Спрощений процес оформлення замовлення знижує кількість покинутих кошиків і підвищує конверсію.

Персоналізація та рекомендації. Використання алгоритмів машинного навчання для персоналізації пропозицій є важливою практикою. На основі історії переглядів і покупок система може пропонувати релевантні товари, що збільшує ймовірність здійснення покупки. Рекомендації можуть включати супутні засоби догляду, новинки або товари зі знижками.

Багатоканальна підтримка користувачів. Надання клієнтам можливості звернутися за допомогою через різні канали — чат, телефон, email, соціальні мережі — підвищує рівень довіри до магазину. Швидка реакція на запити та якісна консультація сприяють покращенню обслуговування.

Інтеграція з платіжними системами та безпека транзакцій. Надійна інтеграція з популярними платіжними сервісами, такими як Visa, MasterCard, PayPal, а також локальними методами оплати, забезпечує зручність для користувачів. Важливим аспектом є також захист персональних даних, використання SSL-сертифікатів і багатофакторної автентифікації.

Оперативне оновлення інформації про наявність товарів і логістику. Точна та актуальна інформація про наявність продукції на складі і строки доставки допомагає уникнути розчарування клієнтів і знижує кількість повернень.

Використання аналітики для прийняття бізнес-рішень. Збір і аналіз даних про поведінку користувачів, продажі та маркетингові кампанії дозволяє адаптувати асортимент, оптимізувати ціноутворення і підвищувати ефективність рекламних активностей.

1.3. Вибір платформи та технологій для реалізації системи

Для розробки інтернет-магазину з продажу доглядової косметики було обрано стек технологій, який забезпечує простоту реалізації, ефективність роботи та можливість подальшого розвитку проекту.

Фронтенд

Інтерфейс користувача реалізовано за допомогою стандартних веб-технологій:

HTML5 — для структурування контенту;

CSS3 — для стилізації і забезпечення адаптивного дизайну;

JavaScript — для реалізації динамічної взаємодії з користувачем, роботи з товарами, кошиком, фільтрацією, аутентифікацією та іншими функціями.

Bootstrap - Використання фрейморку Bootstrap дозволяє значно пришвидшити створення сайти та зменшити обсяг коду завдяки вже створеним елементам та компонентам

Бекенд

Node.js – середовище виконання JavaScript на серверній стороні;

Express.js – зручний веб-фреймворк для Node.js. Він використовується для створення API та обробки запитів

Bscrypt – використовується для хешування паролів;

Jsonwebtoken – для роботи з JWT;

Dotenv – для завантаження змінних середовища;

1.4. Обґрунтувння архітектурного підходу

При створенні інтернет-магазину з продажу доглядової косметики було обрано фреймворк Bootstrap для зручності написання та легкості подальшого розвитку сайту

Клієнтська частина (фронтенд)

Фронтенд реалізований у вигляді веб-інтерфейсу, що працює безпосередньо в браузері користувача. Використання HTML, CSS, JavaScript та Bootstrap забезпечує гнучкий, адаптивний інтерфейс, що відповідає сучасним вимогav. Клієнтська частина відповідає за відображення інформації (товари, інформацію про сайт, кошик, вікна з детальним описом товарів).

Серверна частина (бекенд)

Бекенд реалізований на стороні сервера з використанням Node.js. Він відповідає за обробку запитів від клієнтської частини, керування даними та

реалізацію бізнес-логіки. Використання фреймворку Express.js забезпечує структуру для створення API, маршрутизації запитів та роботи з проміжним програмним забезпеченням. Додаткові бібліотеки, такі як bcrypt для безпечного зберігання паролів, jsonwebtoken для аутентифікації на основі токенів, express-validator для валідації вхідних даних та інші, забезпечують необхідну функціональність для безпечної та ефективної роботи серверної частини застосунку. Бекенд взаємодіє з даними (ймовірно, зберігаються у файлах users.json та orders.json, як видно зі структури проєкту) та надає клієнтській частині необхідну інформацію та функціональність.

Переваги обраного підходу

Простота реалізації — використання знайомих технологій і файлової системи дозволяє швидко створювати прототипи і підтримувати проект.

Зручність підтримки — архітектура дозволяє легко відстежувати та виправляти помилки.

Використання однієї мови (JavaScript) для фронтенду та бекенду: Це спрощує процес розробки, дозволяє команді використовувати спільні знання та потенційно прискорює розробку, оскільки не потрібно перемикатися між різними мовами програмування.

Висока продуктивність (Node.js): Завдяки своїй неблокуючій, подієво-орієнтованій архітектурі, Node.js ефективно обробляє велику кількість одночасних з'єднань, що робить його гарним вибором для веб-серверів та API, які інтенсивно працюють з введенням/виведенням.

Ефективна робота з JSON: Node.js та Express чудово підходять для створення RESTful API, які обмінюються даними у форматі JSON, що є стандартом для взаємодії між фронтендом та бекендом у сучасних веб-застосунках.

Велика екосистема та спільнота: Node.js має найбільший у світі репозиторій пакетів (npm), що надає доступ до тисяч готових бібліотек (як ті, що використовуються в проєкті: bcrypt, jsonwebtoken, cors тощо). Це значно

прискорює розробку, оскільки багато типових завдань вже мають готові рішення.

Таким чином, обрана архітектура відповідає поставленим завданням, забезпечує надійну роботу системи і комфортний досвід користувачам.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНТЕРНЕТ МАГАЗИНУ

2.1. Загальна архітектура системи

Загальна архітектура інтернет магазину побудована на зручності користувача та мінімальній кількості функцій. Вона передбачає зручний інтерфейс та приємний дизайн

1. Клієнтська частина (фронтенд)

Ця частина відповідає за інтерфейс користувача, через який здійснюється вся взаємодія із системою. Основні компоненти клієнтської частини:

- **Інтерфейс каталогу товарів** — відображає список мобільних телефонів, назва продукту, ціна.
- **Спливаюче входу та реєстрації** — дозволяє користувачам створювати облікові записи та входити у систему.
- **Кошик** — містить обрані товари з можливістю редагування, отримання суми та “оформлення замовлення”.

  Серверна частина (бекенд)

  - Ця частина відповідає за обробку бізнес-логіки, керування даними та взаємодію з клієнтською частиною через API. Основні компоненти серверної частини:
  - API для керування користувачами: Обробляє запити, пов'язані з реєстрацією нових користувачів, аутентифікацією (входом) і, можливо, керуванням профілями користувачів. Використовує бібліотеки для безпечного зберігання паролів та генерації токенів доступу.
  - API для товарів: Надає дані про доступні товари (назва, ціна, опис) для відображення в каталозі на фронтенді.
  - API для кошика та замовлень: Обробляє логіку додавання/видалення товарів до кошика, розрахунку загальної суми замовлення та процесу "оформлення замовлення", включаючи збереження інформації про замовлення.

2.2. Розробка інтерфейсу користувача

Одним із ключових завдань при створенні автоматизованої системи продажу мобільних телефонів є забезпечення зручного, інтуїтивно зрозумілого та привабливого інтерфейсу користувача. Враховуючи потреби як нових, так і досвідчених користувачів, інтерфейс спроектовано таким чином, щоб забезпечити швидкий доступ до основних функцій та забезпечити комфортне користування сайтом з будь-якого пристрою.

Основні принципи проектування інтерфейсу:

- **Простота та логічність навігації** — всі ключові елементи розташовані у звичних місцях (меню, каталог, кнопки кошика тощо);
- **Мінімалістичний дизайн** — перевага надається простим кольорам, іконкам, структурованій подачі інформації;
- **Фокус на юзабіліті** — кожна дія супроводжується візуальним або текстовим підтвердженням (додавання в кошик, вхід, оформлення замовлення).

Основні компоненти інтерфейсу:

1. **Головна сторінка**
    - Сторінка з товарами
2. **Каталог товарів**
    - Виведення товарів у вигляді карток;
    - Фото, назва, і ціна товару;
    - Картка з більш детальним описом та кнопкою Додати до кошика
3. **Кошик**
    - Список усіх обраних товарів;
    - Можливість редагування кількості та видалення;
    - Виведення загальної вартості;
    - Кнопка «Оформити замовлення».
4. **Вікно входу/реєстрації**
    - Вхід за логіном та паролем;
    - Реєстрація нового користувача з перевіркою паролю

- Повідомлення про помилки авторизації.

Технічні особливості реалізації

- HTML використовується для структури інтерфейсу;
- CSS — для оформлення зовнішнього вигляду, включаючи адаптивність;
- JavaScript — для динамічної взаємодії: робота з кошиком, фільтрацією, реєстрацією, обробка подій кліків тощо.

Розробка серверної частини (бекенд)

Ключовим завданням при створенні автоматизованої системи продажу мобільних телефонів є розробка надійної та ефективної серверної частини, яка забезпечує обробку запитів, керування даними та реалізацію бізнес-логіки. Бекенд спроектовано таким чином, щоб забезпечити безпечну та швидку взаємодію з клієнтською частиною та ефективне керування інформацією про користувачів, товари та замовлення.

Основні принципи проектування бекенду:

- Надійність та ефективність: Забезпечення стабільної роботи сервера та швидкої обробки запитів.
- Безпека: Захист даних користувачів (особливо паролів) та забезпечення безпечної аутентифікації та авторизації.
- Чітке API: Надання зрозумілого та структурованого інтерфейсу для взаємодії з фронтендом.
- Масштабованість: Можливість обробляти зростаючу кількість користувачів та запитів.

Основні компоненти бекенду (у вигляді API):

- API для користувачів:
  - Реєстрація нових користувачів.
  - Аутентифікація (вхід) користувачів за логіном та паролем.
  - Генерація та перевірка токенів доступу (JWT).
  - Обробка помилок аутентифікації/реєстрації.
- API для товарів:

- Надання списку всіх доступних товарів.
- Надання детальної інформації про конкретний товар.

- API для кошика та замовлень:
  - Обробка логіки додавання/видалення товарів до віртуального кошика (на стороні сервера або взаємодія з даними кошика).
  - Розрахунок загальної вартості товарів у кошику.
  - Обробка процесу "оформлення замовлення" та збереження інформації про замовлення.

Технічні особливості реалізації бекенду:

- Node.js: Використовується як середовище виконання для серверного JavaScript.
- Express.js: Легковаговий веб-фреймворк для Node.js, що надає інструменти для маршрутизації запитів, керування проміжним ПЗ та створення API.
- Бібліотеки: Використовуються спеціалізовані бібліотеки для конкретних завдань:
  - bcrypt для безпечного хешування паролів.
  - jsonwebtoken для реалізації аутентифікації на основі JWT.
  - express-validator для валідації вхідних даних.
  - body-parser для парсингу тіла HTTP-запитів.
  - cors для керування політикою CORS.
  - dotenv для роботи зі змінними середовища.
  - express-rate-limit для обмеження частоти запитів.
  - Зберігання даних: Інформація про користувачів та замовлення, ймовірно, зберігається у файлах (users.json, orders.json), з якими взаємодіє бекенд.

2.2.1. Вимоги до UX/UI

UX (User Experience — користувацький досвід) та UI (User Interface — інтерфейс користувача) є критично важливими складовими розробки сучасних веб-систем. Основна мета — забезпечити простоту, швидкість і приємність

взаємодії користувача з системою продажу мобільних телефонів. У процесі розробки інтерфейсу були сформульовані низка функціональних та нефункціональних вимог, що забезпечують ефективну реалізацію UX/UI.

Основні вимоги до UX (користувацький досвід):

- Швидке виконання цільових дій
- Користувач має змогу з мінімальною кількістю кліків знайти потрібний товар, додати його в кошик і оформити замовлення.
- Простота навігації

2.1.1. Основні вимоги до UX (користувацький досвід)

- **Швидке виконання цільових дій.**

Користувач повинен мати можливість із мінімальною кількістю кліків знайти потрібний товар, додати його до кошика та швидко оформити замовлення.

- **Простота навігації.**

Усі ключові елементи інтерфейсу — каталог, кошик, профіль користувача, фільтри — мають бути чітко позначені, логічно структуровані та легко доступні з будь-якої сторінки сайту.

- **Послідовність інтерфейсу.**

Єдиний стиль та стандартизована поведінка елементів на всіх сторінках сприяють передбачуваності взаємодії, що позитивно впливає на комфорт користувача.

- **Візуальний зворотний зв'язок.**

Усі дії користувача повинні супроводжуватись візуальними сигналами: підсвічуванням активних елементів, повідомленнями про успішне виконання або помилку (наприклад, під час авторизації, реєстрації чи оформлення покупки).

- **Адаптивність дизайну.**

Інтерфейс повинен коректно відображатися та залишатися зручним на будь-яких пристроях — комп’ютерах, планшетах і смартфонах — незалежно від розміру екрана.

- **Доступність (Accessibility).**

Веб-система має враховувати потреби користувачів з обмеженими можливостями: підтримка програм для зчитування з екрана, можливість навігації клавіатурою, відповідний контраст тексту та кнопок.

- **Зрозумілі повідомлення про помилки.**

У разі помилок (наприклад, некоректне введення даних або втрата з'єднання) система має інформувати користувача зрозумілими повідомленнями з поясненням суті проблеми та, за можливості, надати рекомендації для її усунення.

- **Індикація стану завантаження.**

Під час виконання тривалих операцій (наприклад, підтвердження замовлення чи завантаження даних) необхідно показувати індикатори прогресу, щоб користувач розумів, що система працює.

- **Ефективний пошук та фільтрація.**

Необхідно забезпечити зручні та інтуїтивно зрозумілі інструменти пошуку й фільтрації товарів за різними критеріями: ціна, бренд, технічні характеристики тощо. Це дає змогу значно скоротити час на знаходження потрібної позиції.

Основні вимоги до UI (інтерфейсу користувача):

1. **Зрозумілий візуальний стиль**
    - Сайт оформлено в мінімалістичному стилі з чіткою структурою та логічним розташуванням елементів.
2. **Контрастність і доступність**
    - Шрифти, кольори та кнопки мають достатній контраст, що робить інтерфейс доступним для людей з вадами зору.
3. **Уніфіковані компоненти**
    - Використання повторюваних шаблонів для карток товарів, кнопок, заголовків і форм — для зменшення когнітивного навантаження на користувача.
4. **Анімації та мікровзаємодії**
    - Легкі анімації при наведенні, відкритті меню або додаванні товару до кошика — для кращого користувацького досвіду.

2.2.2. Прототипування інтерфейсу

Прототипування є ключовим етапом у процесі розробки користувацького інтерфейсу, оскільки дозволяє візуалізувати загальну структуру веб-додатку, визначити основні функціональні блоки та перевірити логіку навігації між сторінками ще до початку реалізації програмного коду. За допомогою прототипу можна протестувати зручність розміщення елементів інтерфейсу (UI) та оцінити загальну ефективність користувацького досвіду (UX).

Використання прототипів дає змогу:

- своєчасно виявити недоліки у побудові структури сторінок;
- уникнути дублювання функціональності або перенавантаження інтерфейсу;
- забезпечити узгодженість дизайну на всіх сторінках сайту;
- заощадити ресурси під час етапу реалізації, скоротивши кількість змін після написання коду.

Етапи створення прототипу:

1. **Створення каркасних ескізів (Wireframes)**
    - Визначено основні шаблони сторінок: каталог товарів, сторінка товару, кошик, реєстрація/вхід, оформлення замовлення;
    - Встановлено логічну ієрархію блоків: заголовок, меню навігації, контентна зона.
2. **Розробка інтерактивного прототипу**
    - Брав шаблон Figma інтернет-магазину косметики
    - Прототип включав основні дії: вибір товару, додавання до кошика, реєстрацію користувача.
    - Передбачені повідомлення про помилки та підтвердження дій.
3. **Оцінка зручності користування**
    - Проведено внутрішнє тестування макету на предмет зручності інтерфейсу;
    - Внесено правки відповідно до фідбеку: скорочено кількість кроків при оформленні замовлення.

Основні прототиповані сторінки:

- **Каталог товарів** — плитковий вигляд товарів;
- **Картка товару** — Невелике зображення, кнопка «Додати до кошика»;
- **Кошик** — список товарів з цінами та кількістю, кнопка оформлення замовлення;
- **Реєстрація/вхід** — прості форми з полями.

Результати прототипування:

- Усі сторінки макету були протестовані на предмет логічної послідовності та відповідності очікуваному сценарію взаємодії користувача;
- Було отримано повну візуалізацію майбутнього інтерфейсу, що дозволило значно пришвидшити подальший етап верстки;
- Прототип став основою для створення HTML-шаблонів, стилізації через CSS та динаміки за допомогою JavaScript.

2.3. Алгоритми роботи системи

Алгоритми, закладені у функціонування автоматизованої системи продажу мобільних телефонів, визначають послідовність дій користувача та реакцію системи на ці дії. Основна мета — забезпечити логічну, надійну й ефективну обробку запитів з боку користувача, а також гарантувати коректне функціонування ключових процесів: автентифікації, перегляду товарів, додавання до кошика, оформлення замовлення тощо.

2.3.1. Алгоритм реєстрації користувача

1. Користувач відкриває форму реєстрації.
2. Вводить логін, пароль.
3. JavaScript код на сторінці збирає дані з форми.
4. Ці дані відправляються на сервер
5. Система перевіряє:
    - чи заповнені всі поля;
    - чи відповідають умови безпеки (мінімальна довжина паролю тощо);
    - чи логін унікальний (перевірка в базі даних).

6. Якщо всі перевірки пройшли успішно:
    - Дані нового користувача (логін, хешований пароль) додаються до users.json.
    - Користувач отримує повідомлення про успішну реєстрацію або перенаправляється на сторінку входу.
7. Якщо щось не вірно, вилазить помилка.

2.3.2. Алгоритм входу користувача

1. Користувач вводить логін та пароль.
2. Якщо введені дані неправильні — виводиться повідомлення про помилку.
3. JavaScript код на сторінці збирає введені логін та пароль.
4. Ці дані відправляються на сервер
5. На сервері (app.js):
6. Система отримує логін та пароль.
7. Система читає дані користувачів з файлу users.json.
8. Система шукає користувача з таким логіном у даних, прочитаних з users.json.
9. Якщо користувача знайдено:
    - Система порівнює введений пароль (після відповідного хешування, якщо воно використовується) зі збереженим паролем для цього користувача.
    - Якщо паролі збігаються, вхід успішний. Сервер може створити сесію або видати токен.
    - Якщо паролі не збігаються, вхід не вдається.
10. Якщо користувача з таким логіном не знайдено, вхід не вдається.

2.3.3. Алгоритм додавання товару до кошика

Користувач переглядає список товарів на сторінці.

1. На картці конкретного товару користувач натискає кнопку "Додати до кошика".
2. JavaScript код на сторінці перехоплює цю подію натискання кнопки.

3. Цей JavaScript код збирає необхідні дані про товар з відповідної картки (наприклад, ID товару, назву, ціну, початкову кількість - зазвичай 1).

4. JavaScript код отримує поточний стан кошика.

5. JavaScript код перевіряє, чи є вже такий товар у кошику:

- Якщо товар вже є, збільшується його кількість у масиві кошика.
- Якщо товару немає, новий об'єкт товару (з кількістю 1) додається до масиву кошика.

6. Оновлений масив кошика зберігається.

7. JavaScript код динамічно оновлює відображення кошика на сторінці:

- Оновлюється список товарів у кошику, відображаючи доданий товар або змінену кількість.
- Перераховується та оновлюється загальна сума всіх товарів у кошику.

2.4.1. Клієнтська частина (frontend)

Клієнтська частина відповідальна за візуалізацію інтерфейсу користувача та обробку дій користувача у браузері. Реалізована за допомогою наступних технологій:

- **HTML** – структура сторінок;
- **CSS** – стилізація елементів, та адаптивний дизайн;
- **JavaScript** – динамічна взаємодія з DOM, обробка подій, анімації;

Основні компоненти клієнта:

- **Головна сторінка** — відображає товари;
- **Каталог товарів** — перегляд карток товарів;
- **Кошик** — список обраних товарів, підрахунок загальної суми;
- **Форми входу та реєстрації** — перевірка правильності вводу данних

2.4.2. Серверна частина (Backend)

Серверна частина відповідальна за обробку запитів від клієнта, керування даними користувачів та замовлень, а також забезпечення безпеки (автентифікація, валідація). Реалізована за допомогою наступних технологій:

- Node.js: Середовище виконання JavaScript на стороні сервера.
- Express.js: Популярний фреймворк для Node.js, що спрощує створення вебсерверів та API.
- JSON файли (users.json, orders.json): Використовуються для простого зберігання даних користувачів та замовлень замість традиційної бази даних.

Додаткові бібліотеки Node.js:

- fs (File System): Для читання та запису даних у JSON файли.
- bcrypt: Для безпечного хешування паролів користувачів.
- jsonwebtoken (jwt): Для створення та перевірки JSON Web Tokens (JWT) для автентифікації користувачів.
- express-rate-limit: Для обмеження кількості запитів (наприклад, для захисту від перебору паролів).
- express-validator: Для валідації вхідних даних від користувача (наприклад, при реєстрації чи оновленні профілю).
- dotenv: Для завантаження конфігураційних змінних з файлу .env (наприклад, секретного ключа для JWT).

Основні компоненти та функціонал бекенду:

- API Ендпоінти: Набір URL-адрес, які клієнтська частина використовує для взаємодії з сервером (наприклад, /register, /login, /saveOrder, /user-orders, /profile-data, /update-profile).
- Логіка автентифікації: Обробка реєстрації нових користувачів (включаючи валідацію та хешування паролів) та входу існуючих користувачів (перевірка паролів, генерація JWT).
- Авторизація: Перевірка JWT для захищених маршрутів, щоб переконатися, що запит надходить від автентифікованого користувача.

- Керування даними користувачів: Збереження, завантаження та оновлення інформації про користувачів у файлі users.json.
- Керування даними замовлень: Збереження та завантаження інформації про замовлення у файлі orders.json, включаючи зв'язок замовлення з користувачем.
- Валідація даних: Перевірка коректності та безпеки даних, що надходять від клієнта (наприклад, формат телефону, мінімальна довжина паролю).
- Обмеження частоти запитів (Rate Limiting): Захист певних маршрутів (як-от вхід) від надмірної кількості запитів.

Цей опис базується на типовому використанні зазначених технологій та структурі файлів вашого проєкту.

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНА ЧАСТИНА ТА ТЕСТУВАННЯ СИСТЕМИ

3.1. Методика тестування функціональності

Для забезпечення коректної роботи Інтернет магазину з продажу доглядової косметики було проведено функціональне тестування. Основна мета — перевірити відповідність фактичної роботи системи її функціональним вимогам та виявити можливі помилки або нестабільності у роботі.

3.1.1. Мета тестування

- Перевірити коректність працювання сайту у різних браузерах та пристоях
- Коректний вивід інтерфейсу
- Коректна робота серверної частини

3.1.2. Методика тестування

Функціональне тестування виконувалося вручну згідно з підготовленими **тест-кейсами** для кожного з основних модулів системи. Кожен тест-кейс містив:

- Назву функціональності;
- Початкові умови;
- Вхідні дані;
- Очікуваний результат;
- Фактичний результат;
- Статус виконання (успішно / помилка).

3.1.3. Основні сценарії тестування

№	Функціональність	Дії тестувальника	Очікуваний результат	Статус
1	Перехід на сайт	Відкрити головну сторінку сайту у браузері.	Сайт завантажується без помилок, відображається головна сторінка.	✓

2	Реєстрація з порожніми полями	Спробувати зареєструватися, залишивши одне або кілька обов'язкових полів порожніми.	Система виводить повідомлення про помилку, вказуючи, які поля потрібно заповнити.	✓
3	Додавання товару в кошик	Натиснути «Додати до кошику» на товарі	Товар додається до кошика, відображення кошика оновлюється	✓
4	Правильність відображення товару та суми у кошику	Додати різні товари в різній кількості	У кошику відображаються всі додані товари з правильною кількістю та ціною, загальна сума розрахована вірно.	✓
5	Реєстрація з існуючим логіном	Спробувати зареєструватися, використовуючи логін, який вже існує в системі.	Система виводить повідомлення про помилку, що користувач із таким ім'ям вже існує.	✓
6	Вхід з правильними даними	Ввести коректні логін та пароль зареєстрованого користувача у формі входу.	Користувач успішно авторизується.	✓

7	Вхід з невірним паролем	Ввести коректний логін, але невірний пароль у формі входу.	Система виводить повідомлення про помилку.	✓

3.1.4. Середовище тестування

- **Операційна система:** Windows 11
- **Браузери:** Google Chrome, Edge.
- **Технології (frontend):** HTML, CSS, JavaScript, Bootstrap (frontend);
- **Технології (backend):** Node.js, Express.js, JSON file;

3.1.5. Виявлені помилки та їх усунення

У процесі тестування були виявлені такі помилки:

Некоректне выдображення карток товарів: виправлено стилі CSS

Відсутність повідомлення при неправильних данних — **виправлено**: додано перевірку.

Помилка автентифікації через відсутність секретного ключа JWT — виправлено: перевірено наявність та коректність файлу .env у корені проєкту та визначення в ньому змінної JWT_SECRET з унікальним значенням. Забезпечено правильне завантаження змінних середовища при старті сервера.

Висновок:

Після завершення етапу тестування система продемонструвала стабільну та коректну роботу за основними сценаріями використання, включаючи реєстрацію, вхід користувачів, додавання товарів до кошика та перегляд даних. Усі критичні помилки, виявлені в процесі тестування, були оперативно усунені, що значно підвищило надійність додатку.

3.2. Виявлення та усунення помилок

У процесі розробки та тестування інтернет-магазину з продажу доглядової косметики особлива увага приділялася виявленню та своєчасному усуненню помилок. Це дозволило підвищити стабільність та надійність роботи системи.

Процес виявлення помилок включав як автоматизовані перевірки, так і ручне тестування за розробленими сценаріями. Тестування охоплювало ключові функціональні області: інтерфейс користувача (коректне відображення елементів, адаптивність), логіку роботи з кошиком (додавання, видалення, перерахунок суми), процеси автентифікації та авторизації (реєстрація, вхід), а також взаємодію з файловим сховищем даних (збереження та завантаження користувачів та замовлень).

3.2.1. Класифікація виявлених помилок

Всі помилки, знайдені у процесі тестування, були умовно поділені на кілька категорій:

- **Функціональні помилки** — некоректна робота окремих модулів
- **Інтерфейсні помилки** — некоректне відображення елементів, порушення адаптивності або відсутність зворотного зв'язку для користувача.
- **Помилки валідації**: Виникають, коли введені користувачем дані не відповідають встановленим правилам (наприклад, занадто короткий пароль, некоректний формат телефону, спроба реєстрації з існуючим логіном).
- **Помилки даних/сховища**: Проблеми, пов'язані з читанням, записом або цілісністю даних, що зберігаються у файлах.

3.2.2. Приклади виявлених помилок і рішень

№	Опис помилки	Категорія	Рішення	Статус
1	Неможливо змінити кількість товару	Функціональна	Додано кількість того чи іншого товару у кошику	Усунуто
2	Відсутність повідомлення при неправильній формі пошти та паролю	Інтерфейсна	Додано функцію виведення повідомлень про помилки та підсказки	Усунуто

3	Неможливо увійти в систему після перезапуску сервера.	Функціональна	Виявлено, що секретний ключ для JWT не завантажувався коректно з .env. Забезпечено правильне завантаження змінних середовища та перевірку наявності ключа.	Усунуто

3.2.3. Стратегії покращення стабільності

Було реалізовано валідацію форм на клієнтській.

Після кожної зміни в коді виконувалося повторне тестування відповідного функціоналу.

На додаток до клієнтської валідації, всі критичні дані повторно перевіряються на бекенді перед обробкою або збереженням. Це забезпечує цілісність даних та захист від зловмисних запитів, які можуть обійти клієнтську перевірку.

Для захисту від атак типу "грубої сили" та зменшення навантаження на сервер, особливо на маршруті входу (/login), було застосовано обмеження частоти запитів.

Висновок:

Завдяки системному підходу до тестування та відлагодження, який включав категоризацію помилок та застосування стратегій покращення стабільності, вдалося виявити та усунути всі критичні помилки ще до фінального етапу реалізації. Було проведено ретельну перевірку ключових сценаріїв взаємодії користувача з інтерфейсом та бекендом, що дозволило досягти стабільної та передбачуваної поведінки системи.

3.4. Інструкція користувача

Інтерент-магазин з продажу доглядової косметики створено з метою забезпечення зручного та інтуїтивно зрозумілого інтерфейсу, що дозволяє користувачам швидко опанувати всі функції без додаткових зусиль.

- Після відкриття сайту користувач потрапляє на головну сторінку з каталогом товарів. Тут можна легко переглядати доступну продукцію, ознайомлюватися з описом кожного товару, його характеристиками та ціною.
- Додавання товарів до кошика відбувається максимально просто – зазвичай, в один клік на відповідній кнопці в формі товару.
- Робота з кошиком: Перейшовши до кошика, ви побачите список усіх обраних товарів. Тут ви можете легко змінювати кількість кожного товару, видаляти непотрібні позиції, а також бачити автоматичний перерахунок загальної суми вашого замовлення.
- Реєстрація: Якщо ви новий користувач, ви можете швидко зареєструватися, створивши свій обліковий запис. Для цього потрібно заповнити просту форму, вказавши необхідні дані (наприклад, ім'я, телефон та пароль).
- Оформлення замовлення: Коли кошик сформовано, оформлення замовлення реалізовано через просту та зрозумілу форму. Вам потрібно буде підтвердити або ввести контактні дані, необхідні для доставки.
- Системні повідомлення: Усі ваші дії на сайті, будь то додавання товару, реєстрація, вхід чи оформлення замовлення, супроводжуються інформативними повідомленнями. Вони інформують вас про успішне виконання операції або вказують на помилки, якщо такі виникли, допомагаючи швидко зорієнтуватися.

Загалом, система забезпечує комфортний, безпечний та надійний досвід покупки косметики онлайн, задовольняючи потреби широкого кола користувачів завдяки своїй простоті та функціональності.

ВИСНОВКИ

У процесі виконання дипломної роботи було здійснено всебічний аналіз сучасних інтернет-магазинів доглядової косметики, що дало змогу виділити основні функціональні можливості та технологічні тренди у цій сфері. Дослідження кращих практик дозволило визначити ефективні методи побудови користувацького інтерфейсу, організації процесу реєстрації, авторизації, фільтрації товарів та оформлення замовлень.

Обґрунтовано вибір технологій, що відповідають вимогам до гнучкості, швидкості розробки та підтримки системи. Зокрема, було обрано стек технологій із застосуванням HTML, CSS, JavaScript та Bootstrap для клієнтської частини, що забезпечило створення сучасного та адаптивного інтерфейсу. Для серверної частини було обрано Node.js з фреймворком Express.js, що дозволило ефективно реалізувати API та бізнес-логіку. Використання JSON файлів для зберігання даних користувачів та замовлень стало прийнятним рішенням для цілей даного проєкту, забезпечивши простоту реалізації.

Проектування системи виконано із застосуванням модульного підходу, що значно полегшує її супровід і подальше розширення функціоналу. Розроблений інтерфейс відповідає сучасним вимогам UX/UI, забезпечуючи інтуїтивність та зручність користування, а також підтримує адаптивність для різних пристроїв.

Проведено повний цикл тестування системи, що включав функціональне тестування основних модулів, виявлення та усунення помилок. Перевірено коректність роботи роботи з кошиком. В результаті було забезпечено надійну роботу та стабільність системи в різних сценаріях користування.

На серверній стороні успішно реалізовано ключові функції, такі як реєстрація та автентифікація користувачів з використанням безпечного хешування паролів (bcrypt) та генерації JWT для управління сесіями. Впроваджено механізми валідації вхідних даних та обмеження частоти запитів (rate limiting) для підвищення безпеки. Розроблено логіку для додавання товарів до кошика, збереження замовлень у файловому сховищі.

Проведено повний цикл тестування системи, що включав функціональне тестування основних модулів, виявлення та усунення помилок різних категорій (функціональні, інтерфейсні, валідації, даних/сховища). Застосовані стратегії покращення стабільності, такі як валідація на обох сторонах та регресійне тестування, дозволили оперативно виявити та усунути всі критичні дефекти. Перевірено коректність роботи з кошиком, процесами реєстрації, входу та оновлення профілю. В результаті було забезпечено надійну роботу та стабільність системи в різних сценаріях користування.

Отже, створений інтернет-магазин що відповідає поставленим цілям і вимогам, забезпечує рівень зручності, а також має потенціал для подальшого розвитку і масштабування відповідно до потреб бізнесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Bootstrap. Офіційна документація [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs – Дата звернення: 02.04.2025.

2. ChatGPT (GPT-4) – Штучний інтелект від OpenAI [Електронний ресурс]. – Режим доступу: https://chat.openai.com – Дата звернення: 02.04.2025.

3. Laudon, K. C., Traver, C. G. E-commerce 2021: Business, Technology and Society. – 16th ed. – Pearson, 2021. – 912 p.

4. Krug, S. Don’t Make Me Think: A Common Sense Approach to Web Usability. – 3rd ed. – New Riders, 2014. – 216 p.

5. Chaffey, D. Digital Business and E-Commerce Management. – 7th ed. – Pearson, 2019. – 696 p.

6. Shopify Developer Documentation [Електронний ресурс]. – Режим доступу: https://shopify.dev – Дата звернення: 02.04.2025.

7. Stripe. Документація з інтеграції платіжних систем [Електронний ресурс]. – Режим доступу: https://stripe.com/docs – Дата звернення: 02.04.2025.

8. Bitrix24 – CRM-система [Електронний ресурс]. – Режим доступу: https://bitrix24.ua – Дата звернення: 02.04.2025.

9. Ricci, F., Rokach, L., Shapira, B. Recommender Systems Handbook. – Springer, 2015. – 1003 p.

10. Norman, D. A. The Design of Everyday Things. – Revised and Expanded ed. – MIT Press, 2013. – 368 p.

11. Google Developers. Progressive Web Apps [Електронний ресурс]. – Режим доступу: https://web.dev/progressive-web-apps/ – Дата звернення: 02.04.2025.

12. Node.js Documentation [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/docs – Дата звернення: 02.04.2025.

13. Mozilla Developer Network. HTML, CSS, JavaScript Docs [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org – Дата звернення: 02.04.2025.

14. Richardson, L., Ruby, S. RESTful Web Services. – O’Reilly Media, 2007. – 454 p.

15. Figma. Спільнота UI/UX дизайнерів [Електронний ресурс]. – Режим доступу: https://www.figma.com/community/ – Дата звернення: 02.04.2025.

ДОДАТКИ

Додаток А Скріншоти існуючих рішень

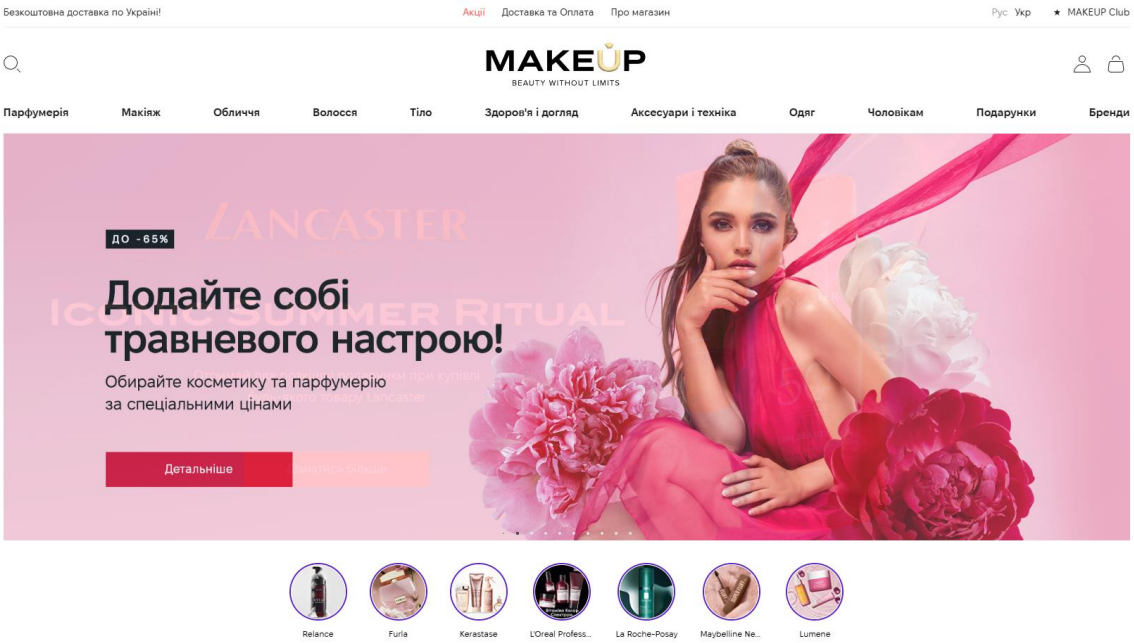

Рисунок А.1 – Головна сторінка інтернет-магазину MakeUp.ua

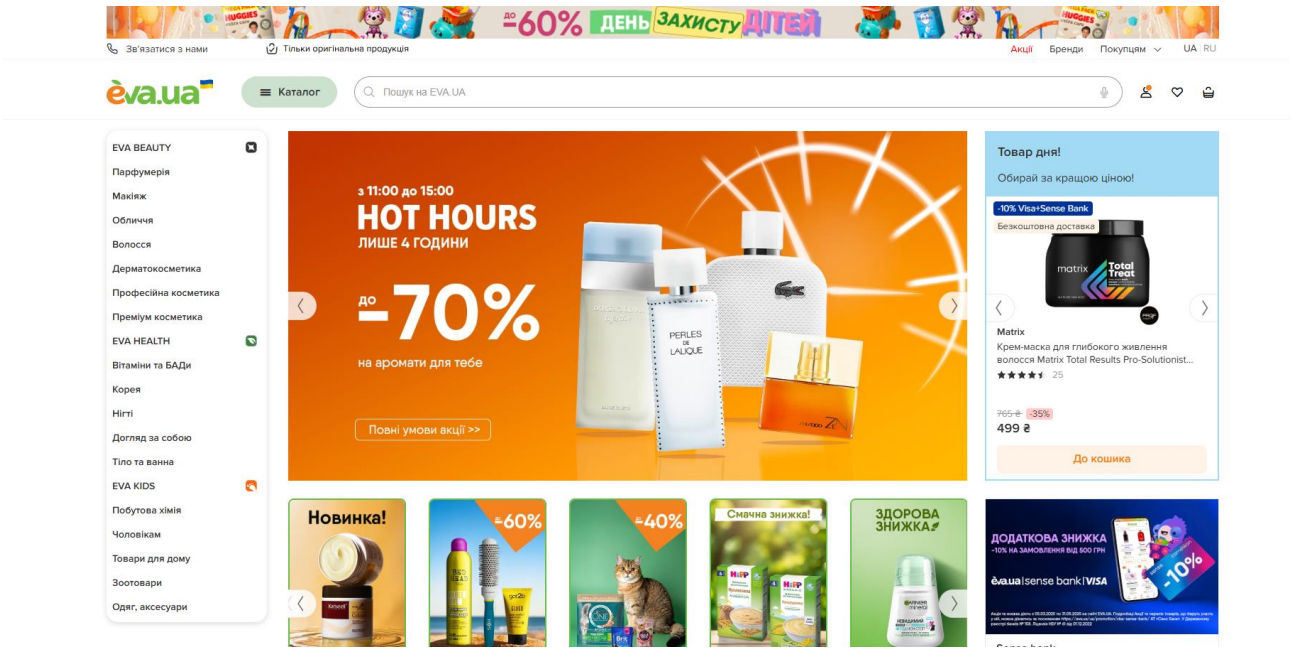

Рисунок А.2 – Каталог товарів з фільтрацією (Eva)

Додаток Б. Скріншоти розробленого інтерфейсу

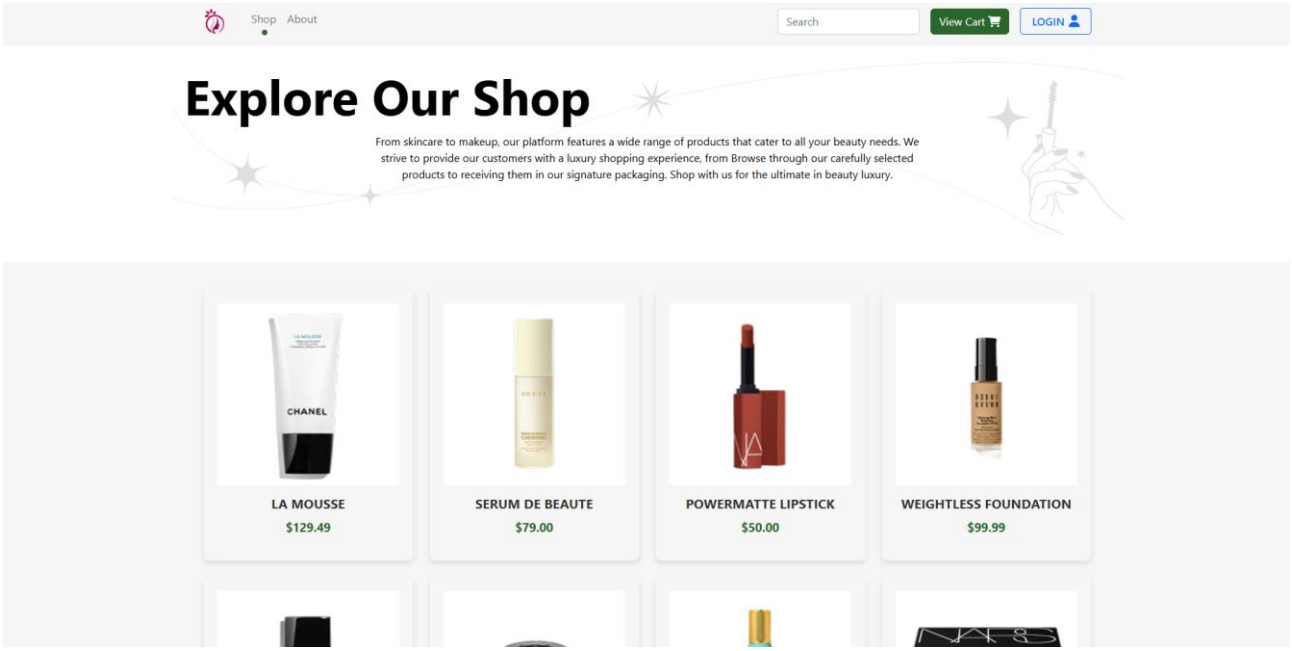

Рисунок Б.1 – Головна сторінка інтернет-магазину

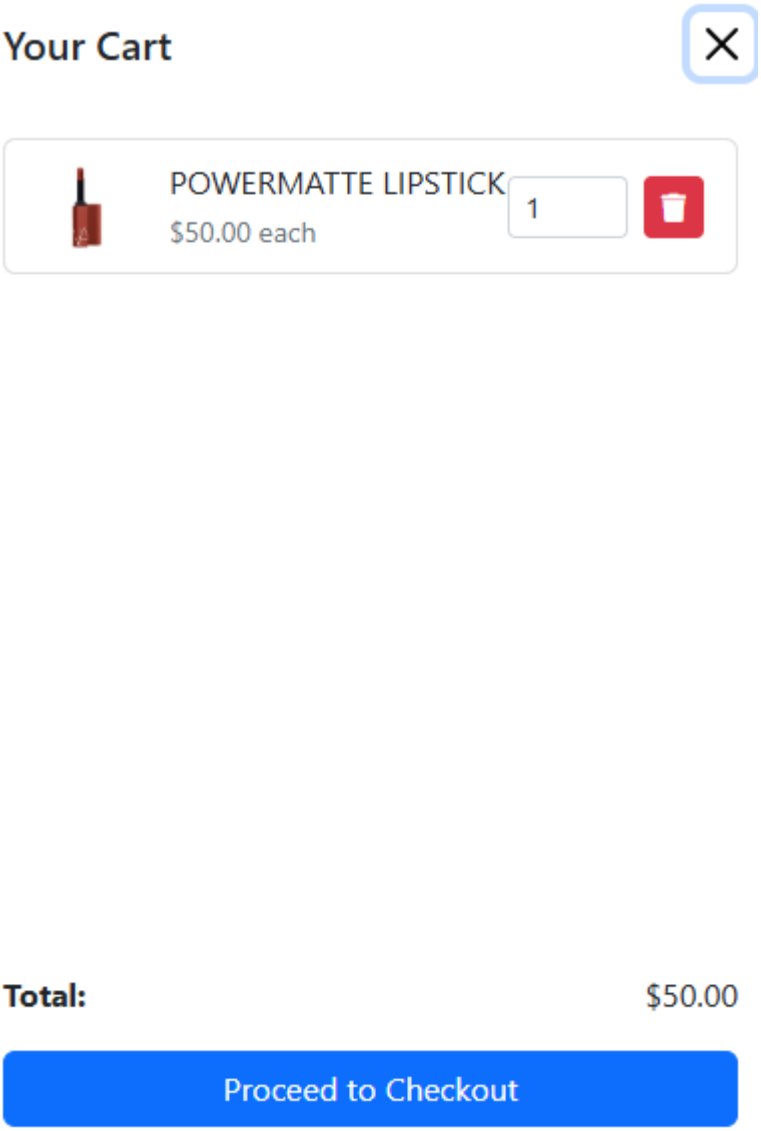

Рисунок Б.2 – Сторінка кошика користувача

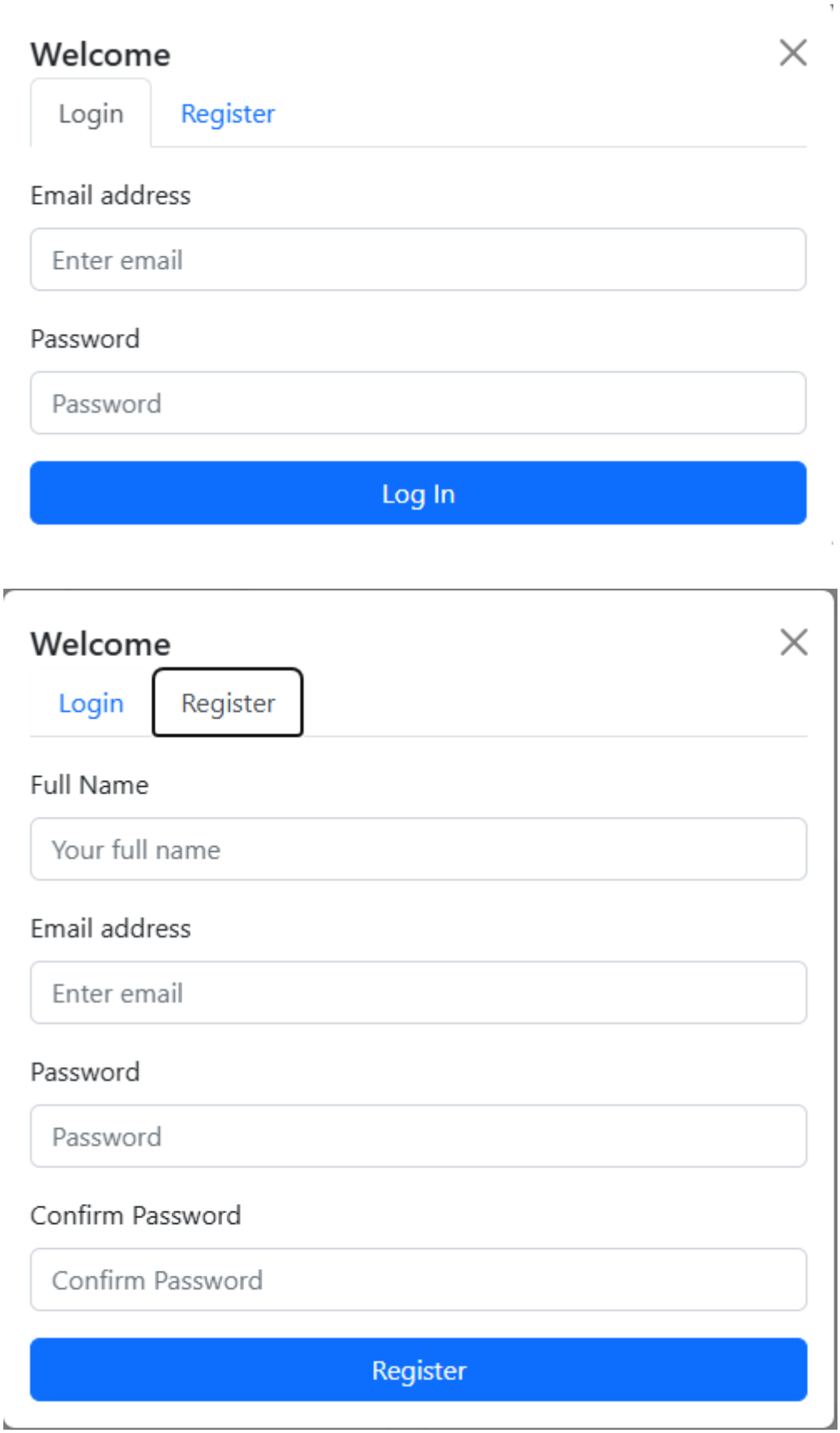

Рисунок Б.3 – Форма реєстрації та входу

Додаток В. Фрагменти коду

В.1 – Кошик

HTML

```
<div
   class="offcanvas offcanvas-end"
   tabindex="-1"
   id="offcanvasCart"
   aria-labelledby="offcanvasCartLabel"
 >
   <div class="offcanvas-header">
    <h5 class="offcanvas-title" id="offcanvasCartLabel">Your Cart</h5>
    <button
     type="button"
     class="btn-close"
     data-bs-dismiss="offcanvas"
     aria-label="Close"
    ></button>
   </div>
   <div class="offcanvas-body d-flex flex-column">
    <ul
     id="cartItems"
     class="list-group mb-3 flex-grow-1 overflow-auto"
     style="max-height: 400px;"
    >

    </ul>
    <div
     class="d-flex justify-content-between align-items-center mb-3"
    >
     <strong>Total:</strong>
     <span id="cartTotalPrice">$0.00</span>
    </div>
    <button id="checkoutBtn" class="btn btn-primary w-100" disabled>
     Proceed to Checkout
    </button>
   </div>
 </div>
```

JS

```
let cart = [];

   function renderCart() {
     const cartItemsContainer = document.getElementById('cartItems');
     const totalPriceElem = document.getElementById('cartTotalPrice');
     const checkoutBtn = document.getElementById('checkoutBtn');

     cartItemsContainer.innerHTML = '';

     if (cart.length === 0) {
       cartItemsContainer.innerHTML =
         '<li class="list-group-item text-center text-muted">Your cart is empty.</li>';
       totalPriceElem.textContent = '$0.00';
       checkoutBtn.disabled = true;
       return;
     }

     let total = 0;

     cart.forEach((item, index) => {
       total += item.price * item.quantity;

       const li = document.createElement('li');
       li.className =
         'list-group-item d-flex justify-content-between align-items-center';

       li.innerHTML = `
         <div class="d-flex align-items-center gap-3">
           <img src="${item.image}" alt="${item.name}" style="width: 50px; height:
50px; object-fit: cover; border-radius: 5px;">
           <div>
             <div>${item.name}</div>
             <small class="text-muted">$${item.price.toFixed(
               2
             )} each</small>
           </div>
         </div>
```

```
        <div class="d-flex align-items-center gap-2">
          <input type="number" min="1" value="${
            item.quantity
          }" style="width: 60px;" data-index="${index}" class="form-control form-
control-sm quantity-input"/>
          <button class="btn btn-sm btn-danger remove-btn" data-index="${index}"><i
class="fa fa-trash"></i></button>
        </div>
      `;

      cartItemsContainer.appendChild(li);
    });

    totalPriceElem.textContent = `$${total.toFixed(2)}`;
    checkoutBtn.disabled = false;

    document.querySelectorAll('.quantity-input').forEach(input => {
      input.addEventListener('change', e => {
        const idx = +e.target.dataset.index;
        let val = parseInt(e.target.value);
        if (isNaN(val) || val < 1) val = 1;
        e.target.value = val;
        cart[idx].quantity = val;
        renderCart();
      });
    });

    document.querySelectorAll('.remove-btn').forEach(btn => {
      btn.addEventListener('click', e => {
        const idx = +e.target.closest('button').dataset.index;
        cart.splice(idx, 1);
        renderCart();
      });
    });
  }

  document
    .getElementById('addToCartBtn')
    .addEventListener('click', () => {
```

```
      const name = modalTitle.textContent;
      const image = modalImage.src;
      const priceText = modalPrice.textContent;
      const price = parseFloat(priceText.replace('$', ''));

      const existingIndex = cart.findIndex(item => item.name === name);
      if (existingIndex > -1) {
        cart[existingIndex].quantity += 1;
      } else {
        cart.push({ name, image, price, quantity: 1 });
      }

      renderCart();

      const offcanvasCart = new bootstrap.Offcanvas(
        document.getElementById('offcanvasCart')
      );
      offcanvasCart.show();
    });
  </script>
</script>
```