ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка Web-застосунку для магазину «Flower Lover» мовою JavaScript з використанням фреймворку Vue.js»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Вікторія НІДЗЕЛЬСЬКА
(підпис)

Виконав: здобувачка вищої освіти групи ПД-41
Вікторія НІДЗЕЛЬСЬКА

Керівник: Олена НЕГОДЕНКО
к.т.н., доцент

Рецензент: _____

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Нідзельській Вікторії Русланівні

1. Тема кваліфікаційної роботи: «Розробка Web-застосунку для магазину «Flower Lover» мовою JavaScript з використанням фреймворку Vue.js»
керівник кваліфікаційної роботи к.т.н., доцент Олена НЕГОДЕНКО,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» лютого 2024 р. № 36.

2. Строк подання кваліфікаційної роботи «28» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: технічна документація з описом доступних фреймворків для розробки web-застосунків, опис методів впровадження засобів електронної комерції, науково-технічна література.

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

1. Ознайомлення з предметною областю, огляд та аналіз існуючих застосунків для продажу квітів.
2. Проектування web-застосунку для продажу квіткових композицій.
3. Програмна реалізація та опис функціонування застосунку для продажу квіткових композицій.
4. Тестування застосунку.

5. Перелік графічного матеріалу: *презентація*

1. Аналіз аналогів.
2. Вимоги до програмного забезпечення.
3. Програмні засоби реалізації.
4. Діаграма варіантів використання.
5. Діаграма шляху користувача.
6. Схема архітектури застосунку.
7. Діаграма послідовності
8. Екранні форми.
9. Апробація результатів дослідження

6. Дата видачі завдання «28» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури	28.02.-06.03.2024	
2	Аналіз та дослідження існуючих аналогів	07.03-13.03.2024	
3	Розробка функціональних та нефункціональних вимог до застосунку	14.03-18.03.2024	
4	Проектування архітектури застосунку для продажу квіткових композицій	19.03-25.03.2024	
5	Програмна реалізація застосунку	26.03-20.04.2024	
6	Тестування застосунку	21.04-28.04.2024	
7	Оформлення роботи: вступ, висновки, реферат	29.04-05.05.2024	
8	Розробка демонстраційних матеріалів	06.05-12.05.2024	
9	Попередній захист роботи	13.05-31.05.2024	

Здобувачка вищої освіти _____ Вікторія НІДЗЕЛЬСЬКА
(підпис)

Керівник
кваліфікаційної роботи _____ Олена НЕГОДЕНКО
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 53 стор., 3 табл., 32 рис., 23 джерела.

Мета роботи – автоматизація бізнес-процесів у сфері продажу квіткових композицій шляхом застосування web-застосунку, створеного мовою Javascript з використанням фреймворку Vue.js.

Об'єкт дослідження – процес продажу квіткових композицій в онлайн магазині.

Предмет дослідження – програмне забезпечення для продажу квіткових композицій.

Короткий зміст роботи: У роботі проаналізовано особливості продажу квіткових композицій та виклики, котрі стоять перед підприємцями у цій сфері. Проаналізовано українські веб-застосунку для продажу квітів: Dicentra, Don Pion, Камелія, Lepestki. Розроблено архітектуру застосунку та програмно реалізовані ключові функціональні можливості, зокрема: фільтрація товарів за трьома характеристиками: колір, формат, ціна, додавання та видалення товарів з кошика, редагування кількості композицій у замовленні, огляд вмісту кошику та підрахунок суми до сплати, можливість залишити відгук або запитання. Проведено два види тестування застосунку: сумісності та функціональне. Для розробки веб-застосунку було обрано JavaScript з використанням Node.js для Backend частини та Vue.js для Frontend розробки. Сферою використання застосунку є автоматизація бізнес-процесів для продажу квіткових композицій.

КЛЮЧОВІ СЛОВА: ОНЛАЙН-МАГАЗИН, ПРОДАЖ КВІТІВ, КВІТКОВІ КОМПОЗИЦІЇ, ДОСВІД КОРИСТУВАЧА.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE – Integrated Drive Electronics, інтегроване середовище розробки

БД – база даних

Моки (mock object) – це повнофункціональні тестові об'єкти, які дають змогу контролювати виклик методів, передачу аргументів і перевірку очікуваної поведінки.

PM - скорочена назва професії Project-менеджера. Роль PM полягає в управлінні проєктами, забезпечуючи їхнє успішне виконання від початку до кінця.

PO - це скорочення від Product Owner, що означає власника продукту. Роль Product Owner полягає у представництві інтересів замовника або користувачів продукту перед розробницьким командою.

BA - це скорочення від Business Analyst, що означає аналітика бізнесу. Роль бізнес-аналітика полягає в аналізі бізнес-потреб, формулюванні вимог до продукту та розробці стратегій для їх впровадження.

ВСТУП

У сучасному світі електронна комерція є невід'ємною частиною бізнес-середовища, зокрема в сфері продажу квітів. Завдяки стрімкому розвитку технологій та зростаючій популярності онлайн-покупок, виникає потреба у зручних та ефективних веб-додатках для спрощення цього процесу.

Мета роботи – автоматизація бізнес-процесів у сфері продажу квіткових композицій шляхом застосування веб-застосунку, створеного мовою Javascript з використанням фреймворку Vue.js.

Об’єкт дослідження – процес продажу квіткових композицій в онлайн магазині.

Предмет дослідження – програмне забезпечення для продажу квіткових композицій.

Для досягнення поставленої мети виділено такі задачі:

1. Дослідити ринок квіткової індустрії та ідентифікувати основних конкурентів. Оцінити їхні веб-платформи та функціонал для розуміння сильних і слабких сторін.
2. Сформулювати вимоги до веб-застосунку, враховуючи функціональність та доступність.
3. Розглянути можливості використання технологій та фреймворків для розробки веб-застосунку.
4. Спроектувати та розробити веб-застосунок для продажу квіткових композицій.
5. Провести тестування функціональності та сумісності.

Реалізація цього проекту вимагає використання передових технологій для розробки необхідного функціоналу та забезпечення позитивного досвіду користувача.

Для розробки веб-сервісу було обрано JavaScript з використанням Node.js для Backend частини та Vue.js для Frontend розробки. Node.js надає високу продуктивність завдяки асинхронному виконанню, дозволяючи ефективно обробляти багато запитів одночасно. Екосистема Node.js також включає широкий вибір модулів та бібліотек, що полегшує розробку та підтримку.

Vue.js в Frontend забезпечує легку розширюваність та декларативний синтаксис для побудови інтерактивних та ефективних інтерфейсів. Можливості Vue.js управління станом додатка та компонентна архітектура сприяють чистоті коду, а реактивність дозволяє ефективно взаємодіяти з користувачем без необхідності оновлення всієї сторінки. Всі ці переваги роблять цей фреймворк відмінним вибором для сучасних та динамічних веб-застосунків.

У процесі розробки веб-застосунку доцільно враховувати принципи створення зручних та приємних для користувачів інтерфейсів. Відповідно до поставленої мети, веб-застосунок спрямований на автоматизацію бізнес-процесів продажу квіткових композицій та надання користувачам зручного інструменту для вибору та придбання квіткових композицій онлайн.

Результатом цієї дипломної роботи є функціональний веб-застосунок, який відповідає вимогам сучасного ринку електронної комерції та автоматизує бізнес-процеси в сфері продажу квітів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧ

1.1 Особливості продажу квіткових композицій та переваги використання веб-застосунків для ведення бізнесу

Сучасний ринок квітів в Україні визначається численними особливостями та викликами, які впливають як на продавців, так і на самих покупців. При розробці цифрових рішень, що впливають на товарний ринок варто звертати особливу увагу на проблемні області ведення цього бізнесу в країні, де буде функціонувати застосунок і чітко визначити, які функції він виконує у поліпшенні бізнес-процесів та задоволенні клієнтських потреб.

Однією з ключових проблем у сфері квіткового бізнесу є потреба в забезпеченні високої якості квітів. Продавці повинні вирішувати завдання збереження свіжості та вигляду квітів, особливо під час транспортування. Це вимагає від них високого рівня досвіду та уваги до деталей.

Ще однією важливою проблемою є змінність цін на квіти, яка залежить від сезону та попиту. Це створює виклики для бізнес-моделі компаній і вимагає систематичного планування цінової політики, щоб підтримувати конкурентоспроможність.

У зусиллях забезпечити свіжість та різноманіття асортименту, продавцям квітів доводиться стикатись з викликом у побудові ефективних партнерських відносин з постачальниками квітів. Це може вимагати значних витрат часу та ресурсів для встановлення надійних і стабільних зв'язків.

Дані проблеми ставлять перед сферою продажу квітів серйозні завдання, і вирішення їх вимагатиме комплексного підходу та стратегічного управління.

Традиційний метод покупки квітів в магазинах призводить до неодноразових його відвідувань клієнтами. Це є фактором, який може стати не завжди зручним для покупців, особливо тих, хто має обмежений час або

перебуває далеку від найближчого магазину в районі. Постійні походи до фізичного магазину можуть створювати додаткові труднощі та обмежити доступність товарів для деяких клієнтів.

У фізичних магазинах клієнтам часто важко отримати достатньо інформації про квіти, зокрема, їх походження та правила догляду. Відсутність досвіду може призвести до невпевненості покупців при виборі та обмежити їхню здатність приймати зважені рішення. Це може впливати на задоволеність клієнтів та їхню готовність повторно обирати даного продавця.

Незручності для клієнтів підкреслюють важливість переходу до більш сучасних та зручних методів придбання квітів, таких як використання онлайн-платформ, що може значно полегшити процес вибору та купівлі квітів для споживачів.

Використання веб-застосунків у сфері продажу квітів надає велику перевагу у забезпеченні зручності та швидкості фінансових транзакцій. Онлайн-платформи дозволяють використовувати комфортні для клієнта системи платежів, що значно зменшує ризик виникнення фінансових проблем та конфліктів між продавцем і клієнтом.

Веб-додатки дозволяють ефективно використовувати цифрові стратегії маркетингу та реклами, зменшуючи витрати на традиційні рекламні методи. Продавці можуть впроваджувати цільовий маркетинг, використовуючи дані про покупців, їхні уподобання та покупкові звички. Це дозволяє підтримувати та залучати нових клієнтів, використовуючи ефективні та економічно доцільні маркетингові стратегії.

Онлайн-магазин дозволяє впроваджувати різноманітні системи лояльності та програми знижок для залучення та утримання клієнтів. Продавці можуть створювати персоналізовані пропозиції, враховуючи історію та вподобання кожного клієнта. Це сприяє підвищенню лояльності споживачів, стимулює повторні покупки та сприяє позитивному іміджу бренду серед конкурентів.

Використання веб-застосунків для оптимізації фінансових операцій, маркетингу та управління лояльністю дозволяє продавцям квітів ефективно конкурувати на ринку та забезпечувати високий рівень обслуговування своїм клієнтам.

Ще однією значущою перевагою є зручність та швидкість процесу покупки. Клієнти можуть здійснювати покупки в будь-який зручний для них час, не виходячи з дому або офісу. Мобільні додатки роблять цей процес ще більш зручним, дозволяючи здійснювати покупки зі смартфона або планшета. Це особливо важливо для зайнятих людей, які цінують ефективність та економію часу.

Рецензії та відгуки користувачів, які доступні на багатьох веб-додатках, стають іншим важливим фактором, який впливає на рішення покупців. Можливість поділитися своїм досвідом та читати відгуки інших дозволяє зробити обдуманий вибір та уникнути неприємних сюрпризів.

Загалом, переваги вибору та покупки квітів через веб-застосунки роблять цей процес більш доступним, ефективним та приємним для сучасних споживачів.

1.2 Функціональні можливості веб-застосунків для продажу квітів

Важливою функцією є можливість оформлення замовлення онлайн. Веб-застосунки надають покупцям зручний інтерфейс для вибору букетів, фільтрації доступних товарів, оформлення замовлення та вибір параметрів доставки.

Веб-застосунки можуть включати функціонал для відстеження статусу замовлення та дати доставки. Це дозволяє покупцям точно визначити момент отримання квітів та вчасно спланувати подарунок. Важливою є і можливість надсилання сповіщень та повідомлень щодо замовлення через мобільні додатки, що полегшує комунікацію між продавцем та клієнтом.

Для покращення взаємодії та залучення клієнтів, веб-застосунки також можуть включати системи лояльності, програми знижок та реалізовувати можливість залишення відгуків. Це стимулює покупців до повторних покупок, розширює аудиторію та формує позитивний імідж бренду.

Загалом, функціональні можливості веб-застосунків для продажу квітів є широким та інноваційним інструментом для оптимізації бізнес-процесів та створення позитивного враження у клієнтів. Їхня ефективна реалізація дозволяє підняти якість обслуговування, розширити ринок та забезпечити успішність у конкурентному світі онлайн-торгівлі квітами.

Споживачі все більше виявляють інтерес до вибору та покупки квітів за допомогою веб-застосунків, тому ринку України існує багато веб-застосунків, спеціалізованих на продажу квітів. Кожен з цих веб-застосунків має свої особливості та функціонал, що робить їх унікальними у конкурентному середовищі. Проте можна виділити спільний для всіх них недолік – не враховані критерії доступності, неадаптивний дизайн та використання неякісного контенту, що негативно впливає на досвід користувачів та знижує їхнє бажання повторно здійснювати покупки через застосунок.

1.2.1 Dicentra

Dicentra – це веб-застосунок, призначений для замовлення та покупки квіткових букетів онлайн. Основною метою веб-застосунку є надання можливості клієнтам, легко знаходити, вибирати та замовляти квіткові композиції.

Dicentra пропонує широкий асортимент квіткових букетів різних видів та кольорів, представлених в каталозі.

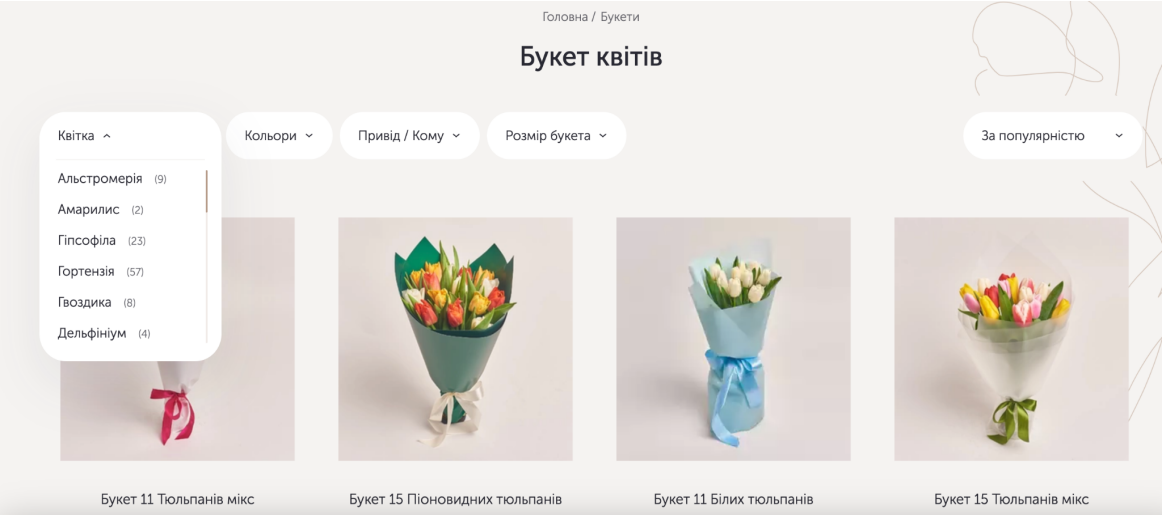


Рис. 1.1 Каталог та фільтри Dicentra

Простий та інтуїтивно зрозумілий алгоритм замовлення дозволяє клієнтам швидко та легко оформити своє замовлення, вказавши адресу доставки та інші необхідні деталі.

Функціональні можливості:

- Можливість перегляду каталогу та фільтрація за різними параметрами.
- Онлайн-консультації та підтримка клієнтів.
- Програма лояльності та знижок для постійних клієнтів.
- Замовлення вибраних товарів.

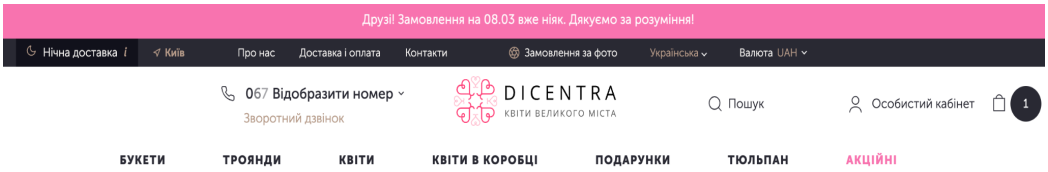


Рис. 1.2 Основне меню Dicentra

Dicentra - онлайн-платформа для покупки квітів, забезпечує широкий вибір товарів та їх фільтрацію. Проте цей веб-застосунок не є якісним, бо при тестуванні знайдено ряд проблем серед яких: низький рівень доступності (рис. 1.3), баг з непрацюючою правою кнопкою слайдера на головній сторінці (рис 1.4).

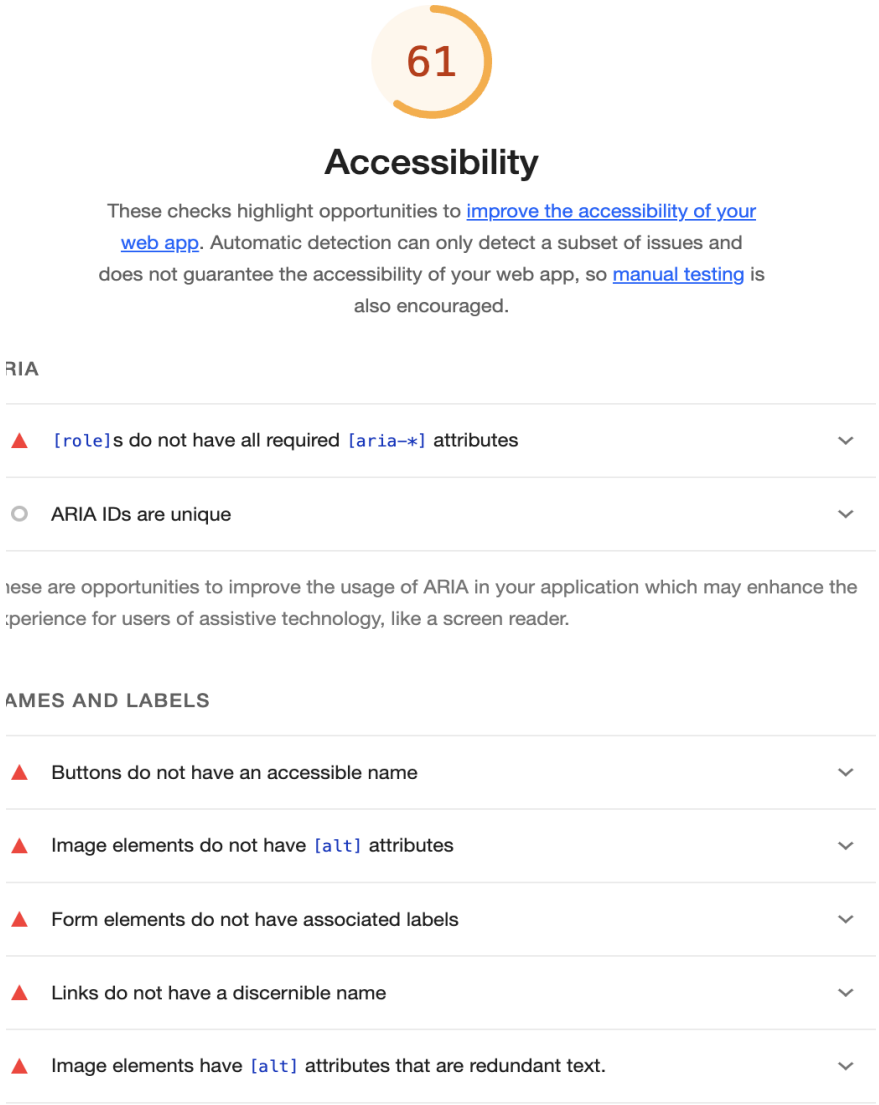


Рис.1.3 Аналіз доступності Dicentra за допомогою розширення Lighthouse

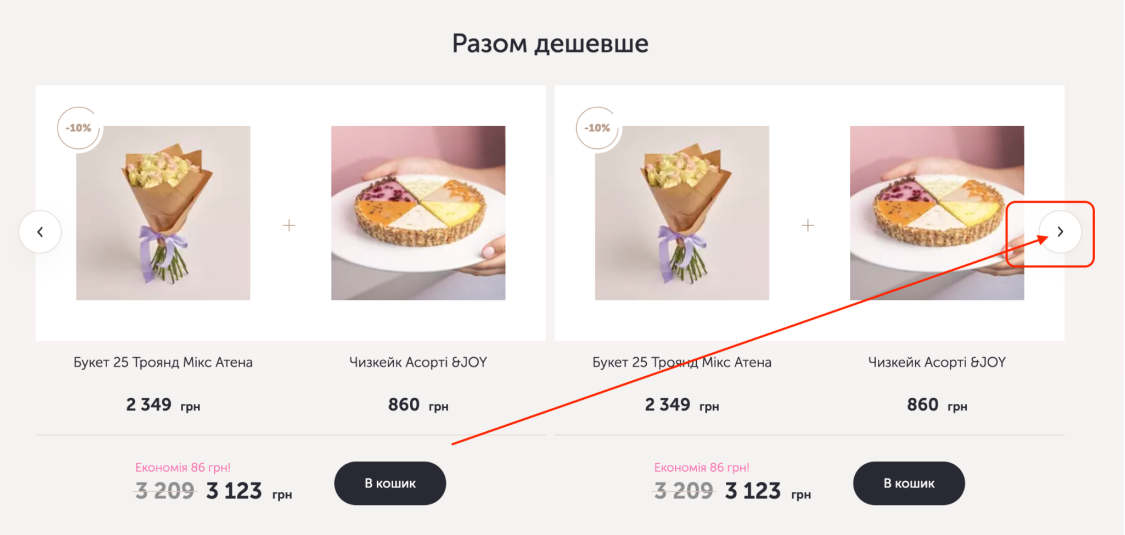


Рис. 1.4 Знайдений баг у Dicentra

1.2.2 Don Pion

Don Pion - це веб-застосунок, спрямований на автоматизацію процесу придбання букетів через інтернет. Основною метою веб-застосунку є забезпечення швидкості у виборі та придбанні квіткових композицій.

Функціональні можливості веб-застосунку включають:

- Можливість перегляду каталогу товарів.
- Замовлення вибраних товарів.
- Використання знижок запропонованих продавцем.

Don Pion представляє собою веб-застосунок, який, на жаль, вирізняється застарілим дизайном та низькою якістю розміщених фотографій. Інтерфейс веб-сайту не відповідає стандартам та не надає користувачеві приємного враження від використання. Фотографії квіткових композицій ускладнюють оцінку товару та знижують довіру покупців шляхом погано підібраних фонів та освітлення. Ці аспекти створюють враження неуважності до деталей в роботі над веб-додатком та псують враження потенційних клієнтів.

Рис. 1.5 Неякісне фото у каталозі Don Pion

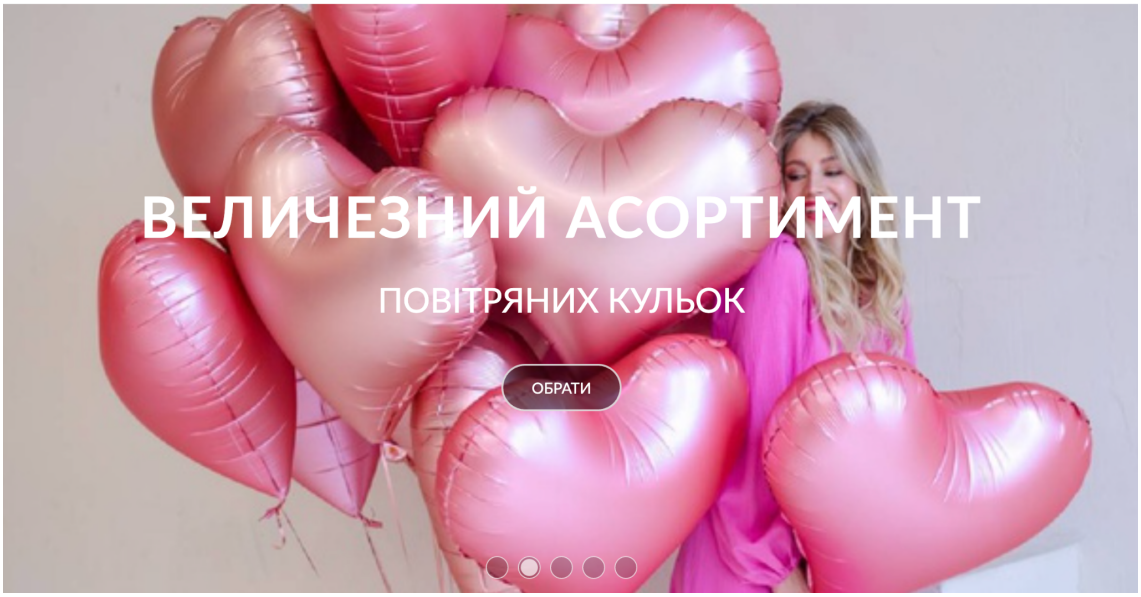


Рис. 1.5 Недостатньо контрасний фон для відображення тексу на слайдері Don Pion

Don Pion має ще один вагомий недолік, а саме той, що він не надає зручний фільтр для каталогу товарів. Відсутність цього елементу ускладнює процес пошуку та вибору товарів для покупців. Замість сучасного та ефективного фільтраційного інтерфейсу, веб-застосунок пропонує меню з безліччю пунктів,

що робить навігацію ускладненою (рис. 1.6). Такий недолік може призвести до втрати зацікавленості покупців до використання веб-застосунку.

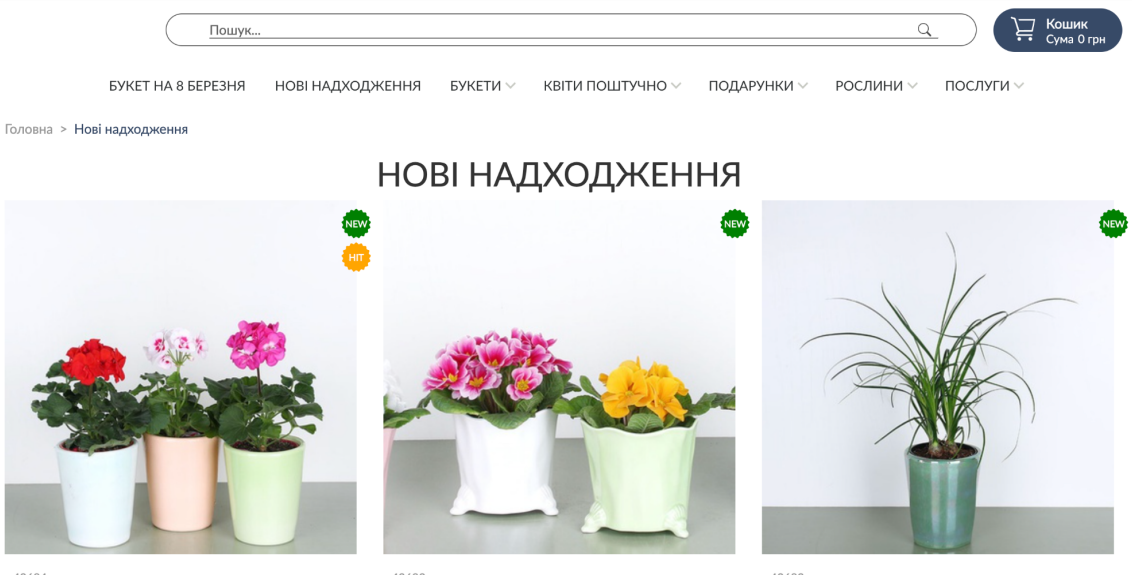


Рис. 1.6 Демонстрація відсутності фільтра у каталозі Don Pion

1.2.3 Камелія

Камелія – це веб-застосунок, який націлено на забезпечення користувачам можливості легко та зручно придбати квіткові букети через інтернет. Орієнтований на високу якість обслуговування та комфорт користувача, проте не покриває всі його потреби, щоб подарувати задоволення від процесу вибору і покупки товарів.

Функціональні можливості веб-застосунку включають:

- Можливість перегляду каталогу товарів.
- Замовлення вибраних товарів.
- Форма зворотнього зв'язку для отримання думок і побажань клієнтів.
- Фільтрація товарів за різними параметрами, включаючи тип квіток, колір, та інші.

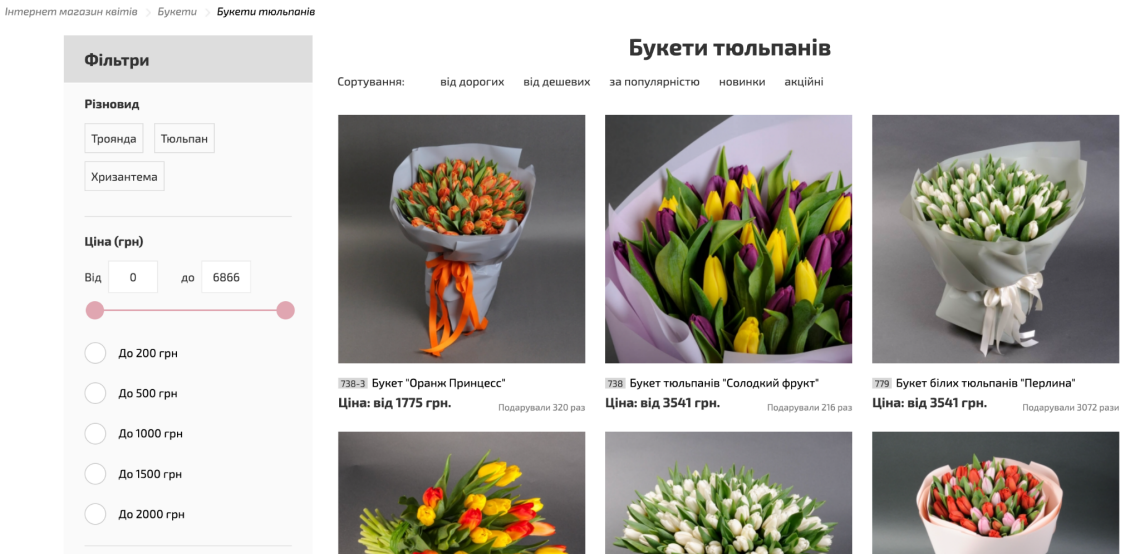


Рис. 1.6 Каталог з фільтрацією у Камелія

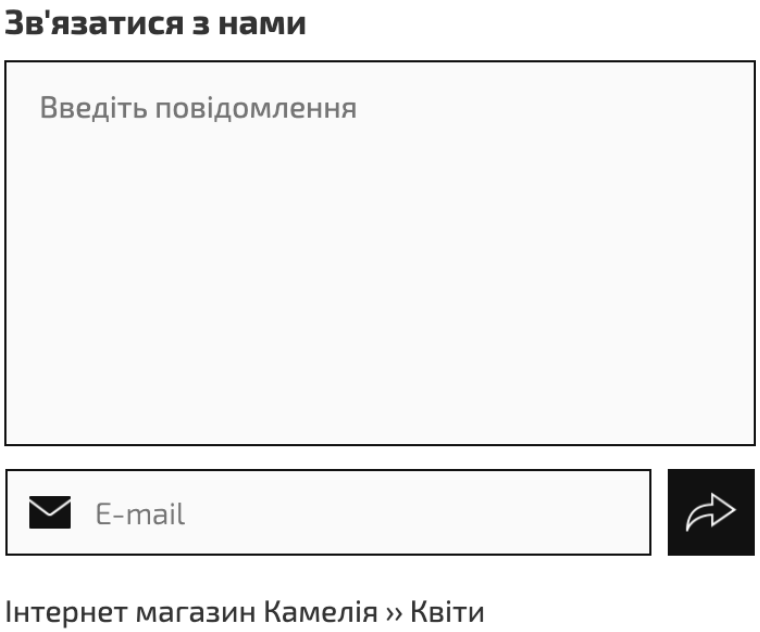


Рис. 1.7 Форма зворотнього зв’язку у Камелія

Швидкість завантаження сторінок та контенту веб-застосунку Камелія є суттєвою перевагою, яка забезпечує користувачам миттєвий доступ до каталогу та інших розділів. Це важливо для забезпечення плавної та ефективної взаємодії з платформою, щоб клієнти могли швидко оглядати асортимент, робити вибір та

оформляти замовлення без зайвого часу та зусиль. Швидкість завантаження створює комфортне та приємне враження від використання додатка, підвищуючи загальну задоволеність користувачів та стимулюючи їхню активність на платформі.

Відсутність чітких та читабельних міток на товарах вважається недоліком веб-додатка Камелія. Неякісне виконання міток або неправильний вибір кольорів може призводити до труднощів у сприйнятті інформації користувачами. Недостатня видимість або нечіткість міток може ускладнювати процес вибору та ідентифікації товарів, що впливає на загальний комфорт користувачів та може викликати негативні враження від взаємодії з платформою. (Рис. 1.8)

Рис. 1.8 Нечитабельна мітка на товарі Камелія

1.2.4 Lepestki

Lepestki - це веб-застосунок для продажу квітів. Основними недоліками цього застосунку є: відсутність єдиного концепту у дизайні, недотримання правил підбору і поєднань кольорів, що може впливати на сердній час використання веб-застосунку користувачами, можливі труднощі у взаємодії із застосунком для користувачів з обмеженими можливостями. Відсутність форми зворотного зв'язку унеможливлює користувачам швидко та зручно надсилати відгуки чи звертатися з питаннями.

Однак, незважаючи на недоліки, більшість стандартних функцій для інтернет-магазину реалізовано і вони коректно відпрацьовують:

- Замовлення вибраних товарів.
- Можливість перегляду каталогу товарів.
- Фільтрація товарів за різними параметрами, включаючи тип квіток, колір, та інші.

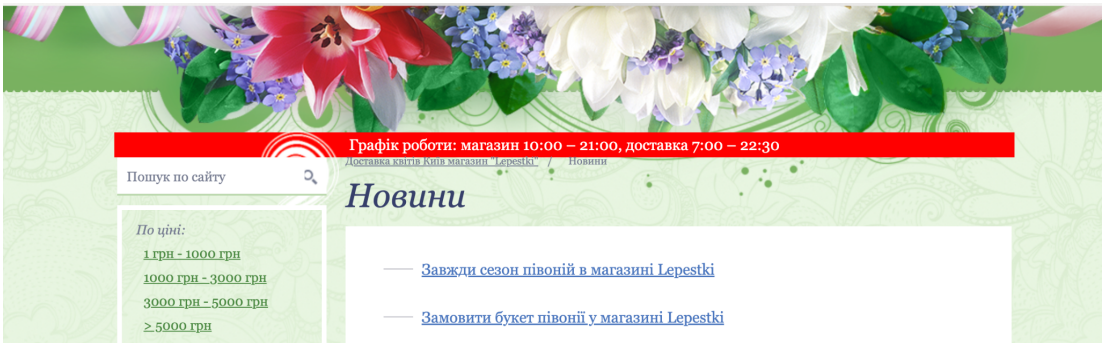


Рис. 1.9 Приклад використання негармонійного поєднання кольорів у Lepestki

У таблиці 1.1 наведено зведену порівняльну таблицю з характеристиками існуючих українських веб-застосунків для продажу квітів. Для збору відгуків про дизайн інтерфейсів розглянутих продуктів було розроблено гугл-форму, яку заповнили 10 користувачів різного віку та статі. Результати опитування наведено на рисунку 1.10.

Позначка часу	Ім'я та прізвище	Оцінка дизайну інтерфейсу Dicentra	Оцінка дизайну інтерфейсу Don Pion	Оцінка дизайну інтерфейсу Lepestki	Оцінка дизайну інтерфейсу Камелія	Оцінка дизайну інтерфейсу Flower Lover
13.03.2024	Родіон Грамушняк	4	3	3	1	5
14.03.2024	Катерина Долюк	3	3	1	3	5
14.03.2024	Оксана Андрієк	4	2	1	3	4
14.03.2024	Марія Прийдун	4	3	2	3	5
14.03.2024	Ірина Косар	4	3	1	3	5
14.03.2024	Анастасія Ковтун	3	4	1	3	5
14.03.2024	Аліна Манджук	4	3	2	4	5
14.03.2024	Боярин Саша	4	3	1	3	5
15.03.2024	Антощук Ілля	4	3	1	3	5
16.03.2024	Сокол Антон	4	3	1	3	5
Середня оцінка		3,8	3	1,4	2,9	4,9

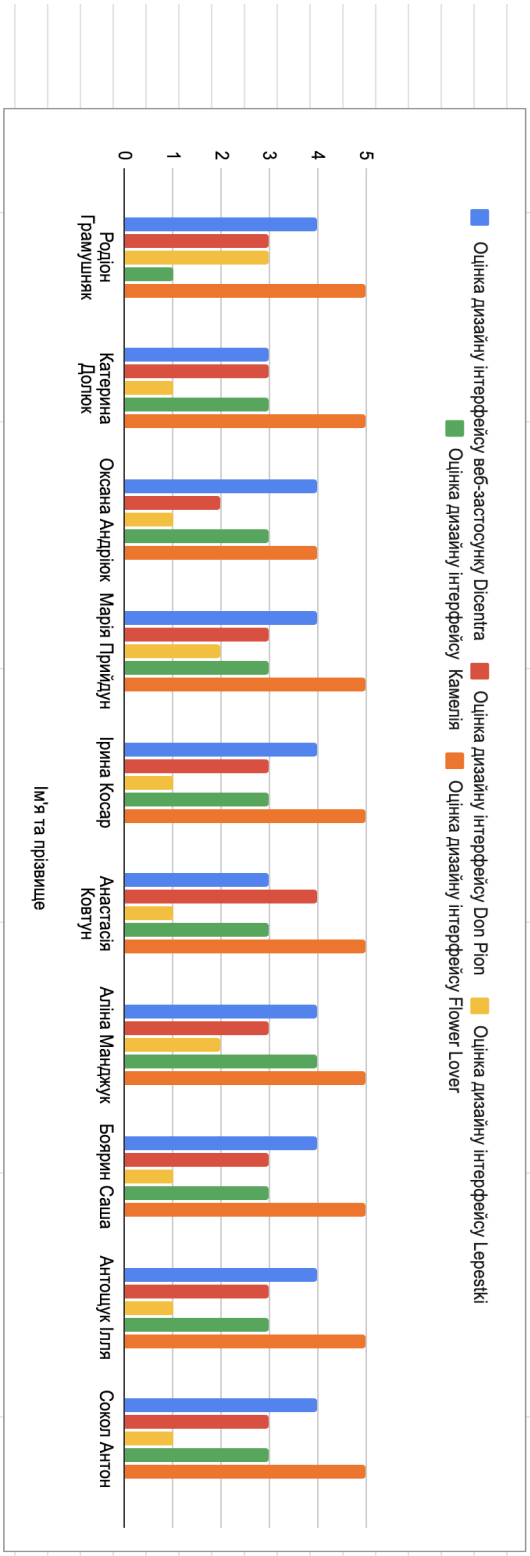


Рис. 1.10 Результати опитування щодо оцінки дизайну інтерфейсів веб-застосунків для продажу квітів

Таблиця 1.1

Зведена таблиця з характеристиками існуючих українських веб-застосунків для продажу квітів

Функціональні та нефункціональні характеристики	**Dicentra**	**Don Pion**	**Камелія**	**Lepestki**	**Flower Lover**
Каталог букетів	+	+	+	+	+
Фільтрація по ціні	+	-	-	+	+
Окрема сторінка товару	+	-	+	-	+
Форма зворотнього зв'язку	+	+	-	-	+
Адаптивність під різні розширення екрану	+	-	+	-	+
Середня оцінка дизайну інтерфейсу від користувачів	3,8	3	2,9	1,4	**4,9**
Швидкість завантаження елементів	2-5 с	5-8 с	2-5 с	5-7 с	**<2 с**

Наведена вище таблиця з функціональними та нефункціональними характеристиками існуючих українських веб-застосунків для замовлення квітів надає корисний огляд доступних на ринку рішень. Вона дозволяє порівняти різні додатки за їх функціональністю, дизайном та іншими факторами.

На основі аналізу таблиці можна зробити такі висновки:

1. Ринок українських веб-застосунків для замовлення квітів є досить конкурентним. Існує багато різних застосунків, які пропонують схожі функції.
2. Найпоширеніші функції включають можливість вибору квітів, оформлення замовлення, фільтрація, зворотній зв’язок через форму.
3. Дизайн застосунків також сильно відрізняється, проте його якість та відповідність стандартам впливає на досвід користувачів.

2 ПОСТАНОВКА ТЕХНІЧНОГО ЗАВДАННЯ І ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ

Цифрове рішення для продажу квіткових композицій розроблено у форматі веб-застосунку.

Веб-застосунок - це програмне забезпечення, доступне через веб-браузер на комп'ютері або мобільному пристрої. Ці додатки використовують веб-технології, для надання користувачам можливостей, подібних звичайним програмам, але з доступом через Інтернет.

Розробка веб-застосунків включає в себе вибір технологій, які найкраще відповідають потребам, наприклад, фреймворки, бази даних, мови програмування та інші технічні аспекти.

2.1 Визначення загальних вимог до системи

Розробка веб-застосунку для продажу букетів у форматі веб-застосунку вимагає ретельного визначення вимог для забезпечення успішного та ефективного функціонування системи.

Після аналізу недоліків та відгуків користувачів про існуючі рішення можна виділити ключові функціональні вимоги для веб-застосунку Flower Lover:

- Користувачі повинні мати можливість переглядати всі доступні букети.
- Має бути надано можливість фільтрувати букети за ціною та типом.
- Застосунок має містити картку з коротким описом та окрему сторінку з детальною інформацією про нього.
- Користувачі повинні мати можливість додавати букети до кошика покупок.
- Кошик покупок повинен відображати назву, зображення, кількість та ціну кожного доданого букета.

- Користувачі повинні мати можливість видаляти букети з кошика покупок.
- Користувачі повинні мати можливість оновлювати кількість букетів у кошику покупок.
- Користувачі повинні мати можливість ввести свою адресу доставки, контактні дані та вибрати спосіб оплати.
- Користувачі повинні мати можливість залишати відгуки про букети та послуги.

Також не менш важливими критеріями успішності проекту є відповідність нефункціональним вимогам представленим нижче:

- Швидкість завантаження сторінок: дві секунди або менше.
- Має бути реалізовано оптимізацію контенту та структури сайту для покращення рейтингу в пошукових системах.
- Використання релевантних метатегів та заголовків сторінок.
- Розробка плану тестування для забезпечення якості та безпомилкової роботи веб-застосунку.
- Застосунок повинен бути адаптивним до різних пристроїв та екранів.
- Веб-застосунок повинен бути доступним для людей з вадами зору.
- Дизайн інтерфейсу повинен забезпечувати позитивний користувацький досвід.

Забезпечення доступності та зручності використання веб-застосунку допомагає розширити аудиторію та поліпшити взаємодію користувачів із цифровим рішенням.

2.2 Вибір технічних засобів реалізації

2.2.1 Вибір середовища розробки

В умовах стрімкого розвитку інформаційних технологій і постійного зростання вимог до програмних продуктів, вибір середовища розробки стає

стратегічним завданням для розробників. Ефективність, продуктивність та зручність роботи над проектом в значній мірі залежать від правильно обраного інструментарію.

Найпоширеніші інтегровані середовищами розробки (IDE) серед фронтенд-розробників наразі є: Visual Studio Code, WebStorm та Sublime Text.

Visual Studio Code (або VS Code) - це інтегроване середовище розробки (IDE), розроблене компанією Microsoft. Це безкоштовне і відкрите програмне забезпечення, призначене для підтримки різних мов програмування та технологій, зокрема фронтенд-розробки. Однією з ключових особливостей VS Code є його легкість використання та висока продуктивність.

VS Code вражає широким спектром розширень, що дозволяє розробникам адаптувати середовище до своїх потреб. Воно підтримує такі мови, як JavaScript, TypeScript, HTML, CSS, Python, Java та інші. Інтеграція з системами контролю версій, вбудований відлагоджувач, автоматичне завершення коду та інші корисні функції роблять VS Code популярним вибором серед фронтенд-розробників для швидкої та зручної роботи над веб-проектами.

WebStorm - це інтегроване середовище розробки (IDE), розроблене компанією JetBrains, спеціалізується на веб-розробці. Воно призначене для фронтенд-розробників, які працюють з JavaScript, HTML, CSS та іншими веб-технологіями. WebStorm надає комплексні засоби для створення високоякісних веб-застосунків.

Однією з ключових переваг WebStorm є його висока продуктивність та потужні можливості відлагодження. Воно підтримує розуміння коду, автоматичне доповнення, відстеження змін, а також інтеграцію з популярними фреймворками, такими як React, Angular і Vue.js. Засоби аналізу та відлагодження сприяють розробці якісного та ефективного веб-коду. WebStorm є комерційним продуктом, але його функціональність і зручність роботи роблять його популярним серед веб-розробників, які цінують високий стандарт розробки.

Sublime Text - це легке та потужне інтегроване середовище розробки (IDE), розроблене компанією Sublime HQ Pty. Завдяки своєму елегантному інтерфейсу та швидкій реакції, Sublime Text завоювало популярність серед розробників, особливо фронтенд-розробників, які цінують простоту та продуктивність у своїй роботі.

Однією з основних особливостей Sublime Text є його розширюваність із застосуванням плагінів та різноманітність можливостей для налаштування. Воно підтримує багато мов програмування та володіє інтуїтивно зрозумілим інтерфейсом, що спрощує роботу з кодом. Sublime Text також володіє широким спектром інструментів для роботи з текстом, включаючи можливості відредагувати кілька рядків одночасно та швидкі клавішні комбінації для ефективної навігації. Завдяки великій кількості розширень та стабільності у роботі, Sublime Text залишається популярним інструментом серед розробників усього світу.

Таблиця 2.1

Порівняльна таблиця IDE для розробки веб-застосунку

Характеристика	**Visual Studio Code**	**WebStorm**	**Sublime Text**
Виробник	Microsoft	JetBrains	Sublime HQ Pty
Мови Підтримки	Значний спектр	JavaScript, HTML, CSS	Значний спектр
Розширення та плагіни	Широкий вибір	Широкий вибір	Широкий вибіри
Умови використання	Безкоштовний	Платний	Безкоштовний або з оплатою за додатковий функціонал
Інтеграція	Системи контролю версій, термінал	Node.js, npm, Git	Системи контролю версій, інтеграція з іншими інструментами
Спільнота та Підтримка	Велика	Велика	Велика

Visual Studio Code обраний через ряд суттєвих переваг, які надають йому конкурентну перевагу в порівнянні з іншими інтегрованими середовищами розробки (IDE). По-перше, VS Code вражає своєю легкістю та ефективністю, що дозволяє розробникам швидко та зручно працювати над веб-проектами. Другою ключовою перевагою є широкий спектр розширень, що підтримують різні мови програмування та технології, надаючи гнучкість у виборі інструментів для конкретного проекту. Враховуючи ці переваги, Visual Studio Code визначається як інтегроване середовище розробки, що найбільше відповідає потребам розробки веб-застосунку у зручності, продуктивності та розширюваності.

2.2.2 Вибір мови програмування

Для розробки веб-сервісів використовуються такі базові інструменти як HTML, CSS та JavaScript або TypeScript.

JavaScript - це мова програмування, яка використовується для розробки фронтенд частини веб-застосунків. Вона має широке застосування в інтернет-розробці та дозволяє створювати інтерактивні та динамічні веб-сторінки.

Хоча TypeScript має свої переваги, такі як статична типізація та більш безпечний код, у даному контексті обрано JavaScript через його відповідність потребам проекту та більшу практичність.

Основні характеристики JavaScript:

Підтримується усіма сучасними веб-браузерами і використовується для взаємодії з користувачем на веб-сторінках, анімації, валідації форм, маніпулювання DOM-елементами та багато іншого.

Може виконуватися не тільки в браузері, але і на сервері, використовуючи такі платформи, як Node.js.

Має вільну (динамічну) типізацію, що дозволяє змінювати типи даних змінних під час виконання програми.

Підтримує функціональні особливості, такі як замикання, анонімні функції, вирази вищих порядків, що дозволяє писати більш компактний та елегантний код.

JavaScript є об'єктно-орієнтованою мовою програмування, де всі дані є об'єктами, а функції - методами цих об'єктів.

Має велику кількість бібліотек та фреймворків, які спрощують розробку веб-застосунків, таких як React.js, Angular.js, Vue.js, jQuery та інші.

JavaScript є однією з найпопулярніших мов програмування у світі та має широкі можливості для розробки різноманітних веб-застосунків.

2.2.3 Вибір фреймворку

Фреймворк – це структурований набір концепцій, стандартів та інструментів, які визначають організацію та роботу програмного забезпечення. Фреймворки забезпечують базові структури та готові компоненти для розробки програм або веб-застосунків. Вони дозволяють розробникам спростити та швидше виконати завдання, надаючи заздалегідь визначені рішення для типових проблем. Використання фреймворків у фронтенд-розробці надає ряд значущих переваг, що сприяють якісному та ефективному створенню веб-застосунків. Ось деякі з ключових переваг:

- Фреймворки надають готові структури та компоненти, що дозволяє розробникам уникати зайвого написання коду з нуля.
- Фреймворки визначають структуру проекту, роблячи його більш організованим та легко зрозумілим.
- Багато фреймворків дозволяють вибирати та розширювати функціоналість за допомогою різних плагінів та модулів. Це робить розробку гнучкою та відповідає різноманітним вимогам проектів.
- Фреймворки включають в себе захисні механізми та норми безпеки, що допомагають у попередженні загроз та атак, таких як введення некоректних даних чи зловживання авторизацією.

У сфері розробки веб-застосунків існує множина фреймворків, серед найпопулярніших з яких можна визначити Angular, Vue.js, та React. У таблиці 2.2 представлено порівняння цих фреймворків.

Таблиця 2.2

Порівняння фреймворків у сфері розробки веб-застосунків

Характеристика	**React**	**Angular**	**Vue.js**
Виражений підхід	Декларативний	Декларативний	Декларативний
Вибірка компонентів	Гнучка, багато готових бібліотек	Вбудована система модулів	Опціональна, можливість вибору
Швидкість	Швидкий рендерінг, віртуальний DOM	Середня швидкість, робота з реальним DOM	Висока швидкодкість, віртуальний DOM
Інтеграція з іншими інструментами	Широкий спектр, висока	Вбудовані засоби, висока	Гнучка, легко інтегрується
Спільнота та підтримка	Велика, активна	Велика, активна	Швидко росте, активна
Вартість	Безкоштовний	Безкоштовний	Безкоштовний

Для розробки веб-застосунку був обраний фреймворк Vue.js.

Vue.js - один з сучасних та популярних фреймворків для розробки веб-застосунків. Заснований на парадигмі компонентного програмування та визнаний за свою простоту та гнучкість, Vue.js швидко завоював популярність серед розробників. Розглянемо основні характеристики та переваги Vue.js у контексті фронтенд розробки.

Vue.js є фреймворком JavaScript, який дозволяє розробникам будувати інтерактивні та динамічні веб-інтерфейси. Однією з ключових рис Vue.js є його легкість вивчення та інтеграція у проекти. Фреймворк надає структуру для

організації коду в компоненти, що спрощує розробку та обслуговування великих веб-застосунків.

Також він володіє прогресивною структурою, що дозволяє розробникам поступово впроваджувати його в існуючі проекти. Великою перевагою є його висока продуктивність та швидкість, завдяки використанню віртуального DOM та ефективному механізму оновлення компонентів.

Переваги Vue.js:

- Привертає розробників своєю простотою та легкістю вивчення. Знання HTML, CSS та JavaScript вже дозволяє швидко стартувати з розробки на Vue.js.
- Фреймворк використовує компоненти для побудови інтерфейсу, що дозволяє створювати перевикористовувані та легко обслуговувані елементи.
- Vue.js можна використовувати поетапно, додаючи його до існуючих проектів. Це робить його гнучким інструментом для розробників.
- Використання віртуального DOM та ефективний алгоритм оновлення роблять Vue.js продуктивним та швидким в роботі з великими обсягами даних.
- Vue.js має велику та активну спільноту розробників, що сприяє обміну досвідом та наданню підтримки.

2.2.4 Вибір бази даних

Для розробки була обрана база даних MongoDB. MongoDB - це потужна документоорієнтована база даних з відкритим вихідним кодом, яка надає гнучкість та масштабованість для різноманітних застосунків.

MongoDB має такі переваги:

Використовує гнучкий формат документів у вигляді JSON-подібних об'єктів, що дозволяє легко зберігати та обробляти дані без потреби відповідності строгій схемі, що характерно для традиційних реляційних баз даних.

MongoDB підтримує горизонтальне масштабування, що дозволяє розподілити дані по кількох серверах, щоб підвищити продуктивність та надійність системи.

Має простий та зрозумілий API, що полегшує розробку та інтеграцію з різноманітними додатками.Завдяки своїй архітектурі та ефективному механізму індексації, MongoDB забезпечує швидкий доступ до даних навіть при великому обсязі інформації.

MongoDB користується популярністю серед розробників, тому в нього є велика та активна спільнота, яка надає швидку підтримку та розвиток інструментів.

2.2.5 Вибір платформи для серверної частини застосунку

Для розробки серверної частини додатка було обрано Node.js. Node.js - це платформа для виконання JavaScript поза браузером, яка дозволяє розробникам створювати ефективні та масштабовані веб-сервери та додатки.

Ось кілька переваг використання Node.js:

Однією з головних переваг Node.js є те, що вона використовує JavaScript, мову програмування, яка вже широко відома та використовується для розробки клієнтської сторони веб-сайтів. Це дозволяє розробникам використовувати ті ж навички та інструменти для розробки як клієнтської, так і серверної частини застосунків.

Він використовує неблокуючу модель введення/виведення (I/O), що дозволяє обробляти багато запитів одночасно без затримок. Це робить Node.js ідеальним вибором для створення високопродуктивних та швидких веб-серверів, які мають велику кількість одночасних підключень.

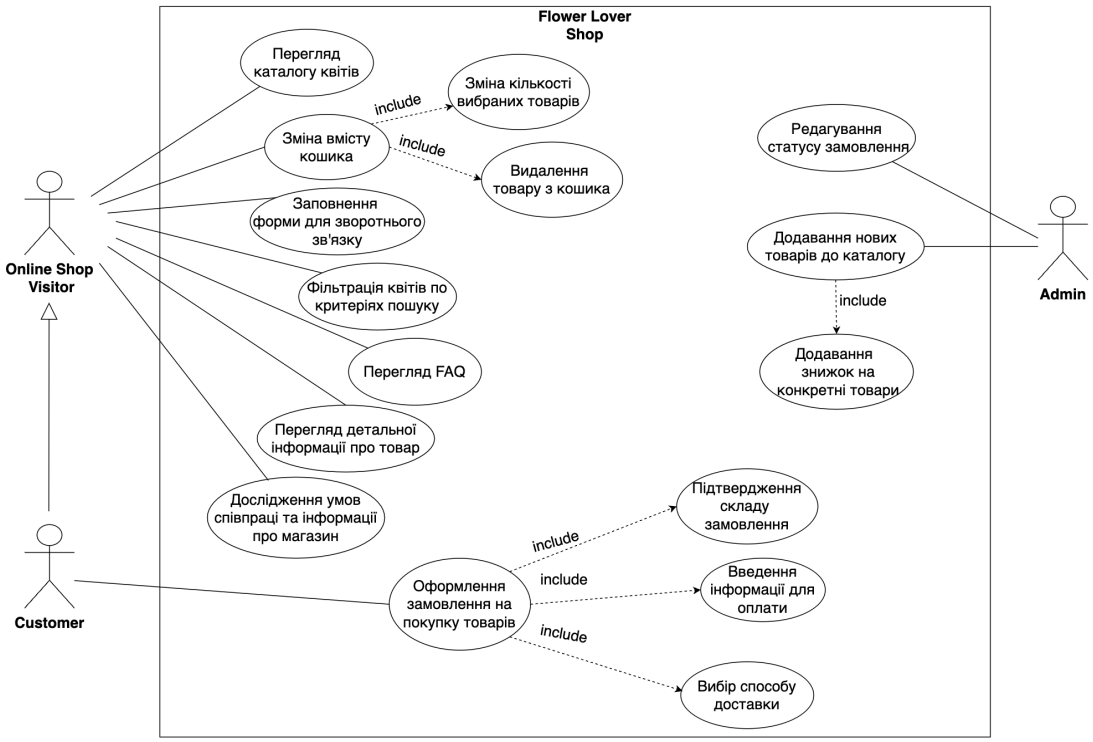
Також платформи добре масштабується, що означає, що ви можете легко розширювати ваш застосунок з додаванням нових серверів та обробкою більшої кількості запитів.

Node.js має велику та активну спільноту розробників, яка постійно розвивається та підтримується. Це означає, що ви можете швидко отримати допомогу та підтримку в разі виникнення проблем або питань.

3 ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБ-СЕРВІСУ

3.1 Визначення груп користувачів

Залежно від своєї ролі в системі, користувач може здійснювати різноманітні дії, які впливають на функціонування системи або дозволяють йому отримувати потрібну інформацію. Ці дії можуть включати пошук та перегляд даних, редагування та збереження інформації, взаємодію з іншими учасниками або виконання конкретних операцій. Приклад такого набору дій можна побачити на ілюстрації 3.1 діаграмі використання, яка показує, як користувачі спілкуються з системою. Це допомагає зрозуміти, як користувачі взаємодіють з системою та які можливості доступні кожному з них, що є ключовим аспектом при розробці ефективного та зручного інтерфейсу.



Рису. 3.1 Use Case діаграма застосунку для продажу квітів

3.2 Проектування внутрішньої архітектури застосунку

Було обрану архітектуру модульного моноліту для розробки застосунку рисунок 3.2 , що передбачає організацію програмного коду у вигляді окремих модулів, що містять у собі всі необхідні компоненти та логіку для виконання певної функціональності.

Основні принципи цього підходу включають:

Розділення на модулі: Весь код додатка розділяється на окремі модулі, які відповідають за різні функціональні частини. Наприклад, можливі модулі для авторизації, управління користувачами, керування контентом тощо.

Компонентна структура: Кожен модуль може містити в собі свої власні компоненти Vue.js, що відображають UI елементи та виконують певні функції.

Стейт-менеджмент: Для управління станом додатка можна використовувати Vuex, який дозволяє зберігати стан додатка та здійснювати зміни в ньому з будь-якої частини програми.

Маршрутизація: Для навігації між різними сторінками додатка використовується Vue Router, який дозволяє визначати шляхи та компоненти, що відображатимуться на кожній сторінці.

Повторне використовування та масштабованість: Модульна архітектура дозволяє підтримувати високу повторну використовуваність компонентів та модулів, а також легко масштабувати застосунок шляхом додавання нових модулів або розширення функціональності існуючих.

Тестування: Кожен модуль можна тестувати окремо, що спрощує процес тестування і забезпечує високу якість коду.

Загалом, підхід до реалізації веб-застосунків на Vue.js як архітектуру модульного моноліту дозволяє забезпечити організацію та підтримку додатка з урахуванням принципів модульності та масштабованості. Архітектуру розробленого застосунку зображено на рисунку 3.3. також на рисунку 3.4 можна простежити послідовність обробки запиту користувача.

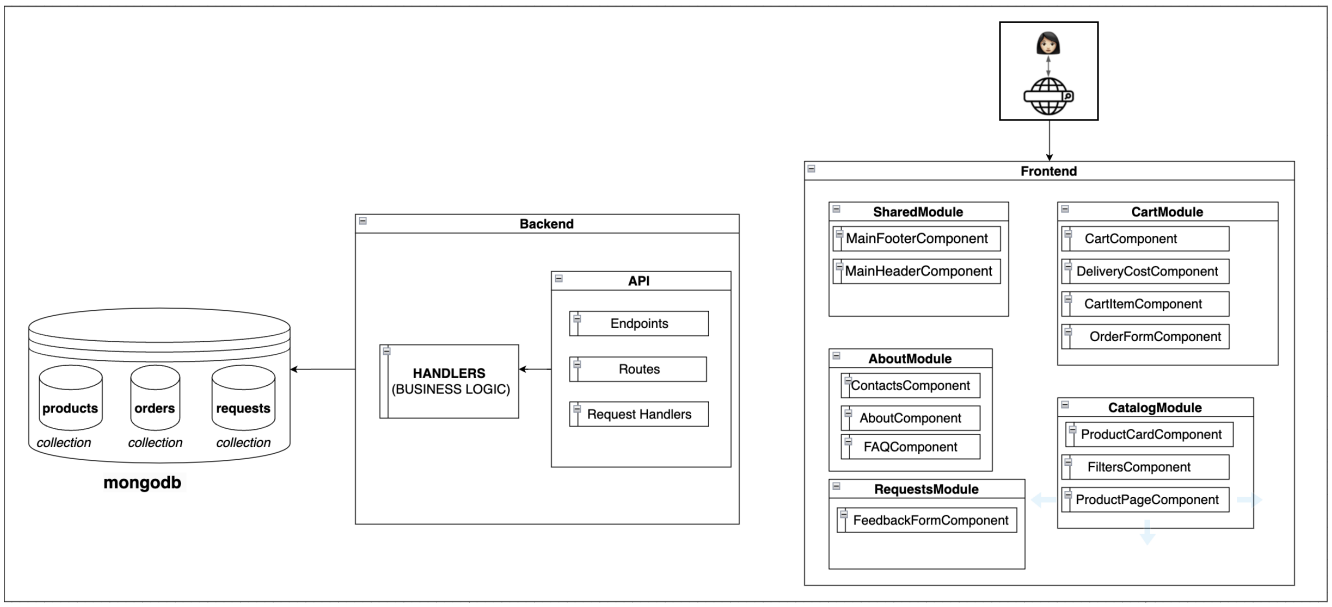


Рис. 3.3 Схема архітектури застосунку

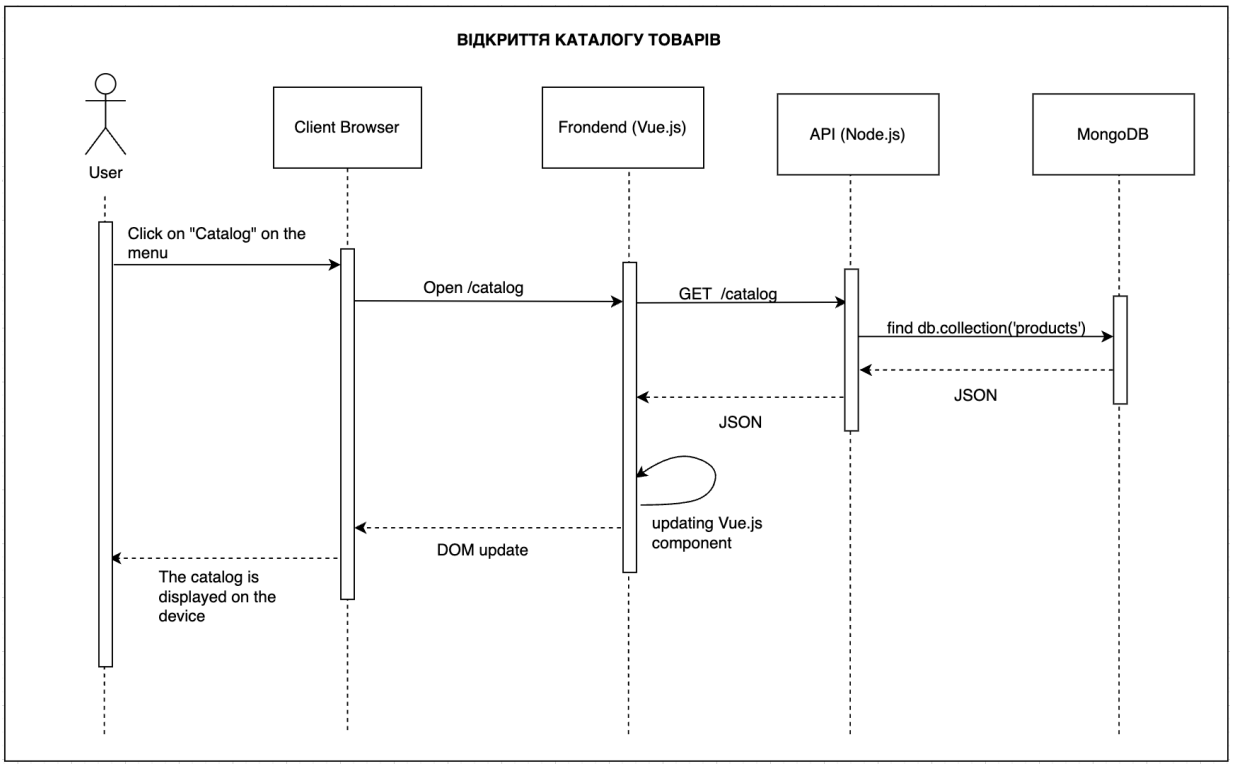


Рис. 3.4 Діаграма послідовності

Маршрутизація Vue Router

Керує URL-адресами та переходами між компонентами. Дозволяє користувачам переходити до різних сторінок магазину, наприклад, до сторінки категорії, сторінки продукту та сторінки кошика.

Back-end (Node.js)

API Node.js написаний на Node.js та використовує Express.js для створення REST API. Надає кінцеві точки для доступу до даних про категорії, продукти та кошик. Взаємодіє з базою даних mango.db для отримання та оновлення даних.

База даних mango.db зберігає дані про категорії, продукти та замовлення.Використовує NoSQL базу даних mango.db для зберігання даних у документно-орієнтованому форматі.

Взаємодія між фронт-ендом та бек-ендом

Фронт-енд Vue.js використовує Axios для надсилання HTTP-запитів до API Node.js. API Node.js використовує mango.db для отримання та оновлення даних. Фронт-енд Vue.js оновлює інтерфейс користувача відповідно до даних, отриманих від API Node.js.

Обробка подій та компоненти у Vue.js

Обробка подій є ключовою концепцією у Vue.js, що дозволяє веб-додаткам реагувати на дії користувача.

Компоненти – це базові блоки будівництва інтерфейсу користувача у Vue.js, які можна повторно використовувати та об'єднувати для створення складних інтерфейсів.

Існує два основних способи обробки подій у Vue.js:

1. Вбудовані обробники використовуються для обробки вбудованих подій DOM, таких як click, mouseover, keydown тощо. Додаються безпосередньо до елементів HTML у шаблоні компонента.

2. Обробники методів використовуються для обробки власних подій

або подій, створених компонентами. Визначаються як методи в екземплярі компонента. Викликаються за допомогою директиви v-on у шаблоні компонента.

Компоненти можуть генерувати власні події, які можуть бути прослухані батьківськими або дочірніми компонентами. Використовуються директиви v-on та emit для генерування та прослуховування подій компонентів.

Модифікатори подій дозволяють додатково налаштувати поведінку обробників подій. Клавіатурні події дозволяють обробляти натискання клавіш на клавіатурі. Використовуються keyup, keydown та інші події.

3.3 Процес підключення MangoDB через Node.js

Процес підключення MangoDB через Node.js реалізовано за наступним алгоритмом:

1. Створення проекту Node.js:

- Створення папки для проекту Node.js.
- Ініціалізація проекту Node.js за допомогою npm init або yarn init.
- Встановлення необхідних залежностей, таких як:

express: Для створення REST API.

mongoose: Для взаємодії з MangoDB.

dotenv: Для безпечного зберігання конфігураційних даних.

2. З'єднання з MangoDB:

Створити файл .env для зберігання конфігураційних даних, таких як URL-адреса MangoDB та секретний ключ. У файлі server.js імпортуваим dotenv та завантажити змінні середовища. Імпортувати mongoose та підключити до бази даних MangoDB за допомогою MONGO_URL з .env.

3. Створити моделі MangoDB:

Створити папку models. У папці models створити файли для моделей, які відповідають документам у MangoDB. Визначити схему Mongoose для кожної моделі, включаючи поля та типи даних. Експортувати моделі.

4. Створення API Node.js:

Створити папку routes.

У папці routes створити файли для маршрутів, пов'язаних з кожною моделлю. Імпортувати відповідні моделі.

Визначити маршрути для отримання даних, створення нових, оновлення та видалення документів.

5. Інтеграція з Vue.js:

У Vue.js компоненті імпортувати axios для надсилання HTTP-запитів.

Використовую axios для надсилання запитів до API Node.js, таких як:

- Отримання даних з API.
- Надсилання даних до API (наприклад, для створення нового відгуку).

```
const express = require('express');
const mongoose = require('mongoose');
const Feedback = require('./models/feedback');
const app = express();
const port = process.env.PORT || 3000;

mongoose.connect('mongodb://localhost:27017/flowerlover', { useNewUrlParser: true,
    useUnifiedTopology: true });

app.use(express.json());
app.post('/api/feedback', async (req, res) => {
  const { name, mobile, comment } = req.body;

  const feedback = new Feedback({
    name,
    mobile,
    comment,
    date: { type: Date, default: Date.now },
  });

  try {
    await feedback.save();
    res.json({ message: 'Feedback saved successfully.' });
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Error saving feedback.' });
  }
});

app.listen(port, () => {
  console.log(`Server listening on port ${port}`);
});
```

Рис. 3.4 Приклад реалізації API Node.js для збереження даних після заповнення форми зворотного зв'язку

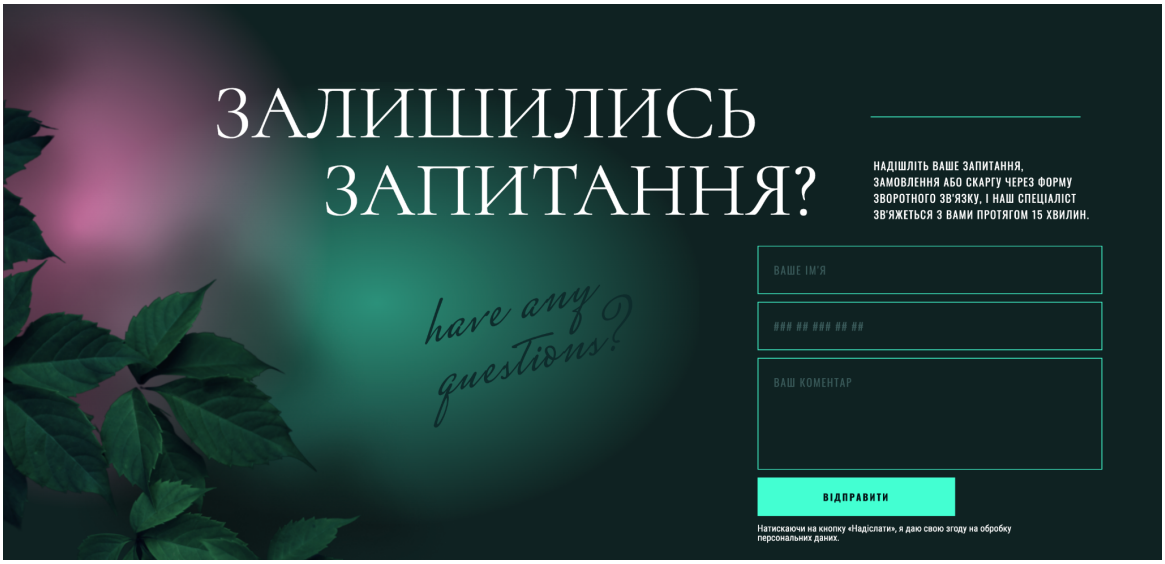


Рис. 3.5 UI форми зворотнього зв’язку

Ця архітектура є не єдиним можливим рішенням для веб-магазину квітів, але вона є хорошою основою для створення потужного, масштабованого та гнучкого веб-застосунку.

3.4 Sass у розробці веб-магазину

Sass - це CSS-препроцессор, який робить CSS більш лаконічним, організованим та динамічним.

Переваги використання Sass:

Використовує змінні, селектори, вкладені правила та міксини, щоб зменшити обсяг повторюваного коду CSS.

Дозволяє створювати та повторно використовувати фрагменти CSS-коду, які називаються міксинами.

Sass дозволяє використовувати змінні для зберігання значень CSS, таких як кольори, шрифти та розміри.

Дає можливість вкладати CSS-правила, що створює більш чітку та ієрархічну структуру коду.

Sass підтримує функції, які дозволяють виконувати математичні операції,

маніпулювати рядками та створювати динамічні стилі.

3.5 LazyLoad: принципи та переваги у веб-магазині

LazyLoad - це техніка оптимізації веб-сторінок, яка відкладає завантаження зображень, відео та інших ресурсів до моменту, коли вони стають видимими користувачеві. Це може значно покращити час завантаження сторінки та загальний досвід користувачів, особливо для веб-магазинів, які зазвичай містять багато зображень.

Принципи роботи LazyLoad:

- LazyLoad замінює зображення та інші ресурси тимчасовими заглушками (наприклад, порожніми контейнерами або зображеннями завантаження).
- Завантаження за потребою. Коли користувач прокручує сторінку і ресурс стає видимим, LazyLoad динамічно завантажує його.
- LazyLoad гарантує, що контент буде доступний користувачам, навіть якщо JavaScript не працює.

Переваги використання LazyLoad у веб-магазині:

- LazyLoad може значно скоротити час завантаження сторінки, що робить веб-магазин більш чутливим для користувачів.
- Зниження використання пропускної здатності. LazyLoad завантажує лише ті ресурси, які дійсно потрібні користувачеві, що економить пропускну здатність та дані.
- Google рекомендує використовувати LazyLoad для оптимізації веб-сторінок, тому його використання може покращити позиції веб-магазину в пошуковій видачі.
- Швидкий час завантаження та плавний досвід користувачів можуть призвести до кращої конверсії та збільшення продажів.

LazyLoad - це потужний інструмент, який може значно покращити

продуктивність та користувацький досвід веб-магазинів. Його просте впровадження робить його цінним активом для будь-якого веб-розробника, який прагне оптимізувати свій веб-магазин.

3.6 Розробка дизайну екранів додатка

Інтернет-магазин квітів Flower Lover прагне надати своїм клієнтам зручний та приємний досвід покупки онлайн. Для цього необхідні чітко продумані та візуально привабливі екрани, які проведуть користувачів крізь процес вибору та замовлення квітів.

Для розробки екранів магазину Flower Lover було використано Figma - популярний інструмент для дизайну інтерфейсів користувача. Figma дозволяє створювати прототипи екранів, тестувати їх та співпрацювати з командою розробників.

Інструменти, які використовувались для створення макета застосунку:

Фрейми: Фрейми - це основні будівельні блоки макетів Figma. Їх можна використовувати для створення окремих екранів, компонентів або груп компонентів.

Авторозміщення: Figma пропонує автоматичні опції розміщення для елементів макета, такі як вирівнювання, розподіл та групування.

Смарт-гіди: Смарт-гіди допомагають точно розмістити елементи макета, використовуючи лінії сітки, магніти та інші візуальні підказки.

Компоненти: Компоненти - це багаторазово використовувані елементи макета, які можна створювати та зберігати. Це економить час і забезпечує послідовність дизайну.

Векторні інструменти: Figma пропонує широкий спектр векторних інструментів для створення та редагування форм, ліній та кривих.

Текстові стилі: Figma дозволяє створювати та зберігати текстові стилі, які можна застосовувати до різних елементів макета.

Зображення та піктограми: Figma дозволяє імпортувати та використовувати зображення та піктограми у макетах.

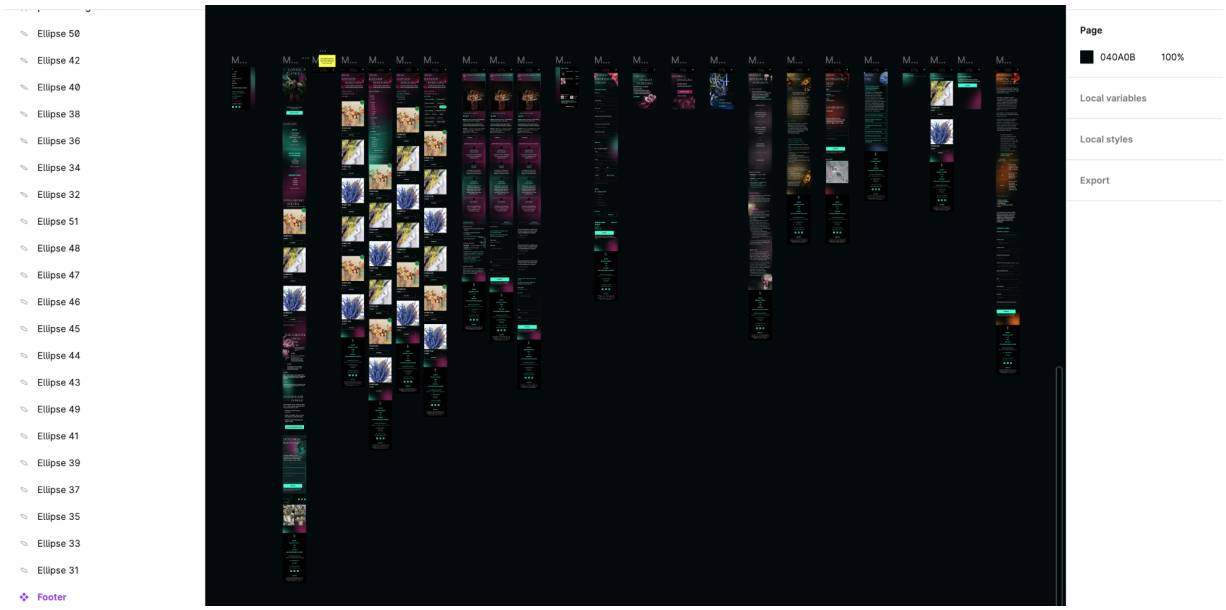


Рис. 3.6 Розроблені екрани застосунку

Використання цих інструментів Figma дозволило створити чітко продумані та візуально привабливі екрани для магазину Flower Lover. Ці екрани допоможуть користувачам легко та зручно знаходити та замовляти квіти онлайн.

Розробка ефективного веб-магазину, який залучає та утримує клієнтів, потребує ретельного планування та розуміння поведінки користувачів. Саме тут на перший план виходить юзер флоу діаграма (UFD) – інструмент, який візуалізує шлях користувача на сайті, від моменту його відвідування до здійснення покупки або виконання іншої цінної дії.

Створення UFD перед розробкою веб-магазину має низку суттєвих переваг:

1. Покращує розуміння користувача. UFD дозволяє розробникам та дизайнерам чітко уявити, як користувачі взаємодіють з сайтом, які кроки вони роблять, де можуть виникнути труднощі або сумніви. Це розуміння дає можливість оптимізувати інтерфейс та усунути потенційні бар'єри на шляху до конверсії.

2. Підвищує зручність користування, допомагає виявити незручні або не логічні елементи інтерфейсу, які можуть негативно вплинути на досвід

користувача. Завдяки цьому UFD сприяє створенню більш інтуїтивно зрозумілого та зручного веб-магазину, що веде до кращого задоволення клієнтів та лояльності.

3. Сприяє командній роботі, слугує спільним мовою для розробників, дизайнерів, менеджерів проектів та інших зацікавлених сторін. Це візуальне представлення процесу покупки допомагає всім учасникам команди чітко розуміти цілі та очікування, а також узгоджувати свої дії для досягнення кращого результату.

4. Полегшує тестування та вдосконалення. UFD може використовуватися як основа для створення тестових сценаріїв, що дає можливість виявити та усунути помилки на ранніх стадіях розробки. Це економить час та ресурси, а також гарантує, що веб-магазин буде функціонувати безперебійно.

5. Підтримує масштабування та може бути легко оновлена та доповнена у міру розвитку веб-магазину та додавання нових функцій. Це робить її цінним інструментом для підтримки довгострокової життєздатності та адаптивності веб-магазину.

Розробка ефективного веб-магазину, який залучає та утримує клієнтів, потребує ретельного планування та розуміння поведінки користувачів. Саме тут на перший план виходить юзер флоу діаграма (UFD).

Створення UFD перед розробкою веб-магазину є важливим кроком, який гарантує зручність користування, покращує конверсію та сприяє довгостроковому успіху вашого онлайн-бізнесу. Вона слугує цінним інструментом для візуалізації та оптимізації шляху користувача, забезпечуючи позитивний досвід для клієнтів та максимізуючи шанси на успіх у динамічному світі електронної комерції. Діаграма шляху користувача для магазину квітів Flower Lover представлена на Рисунку 3.7.

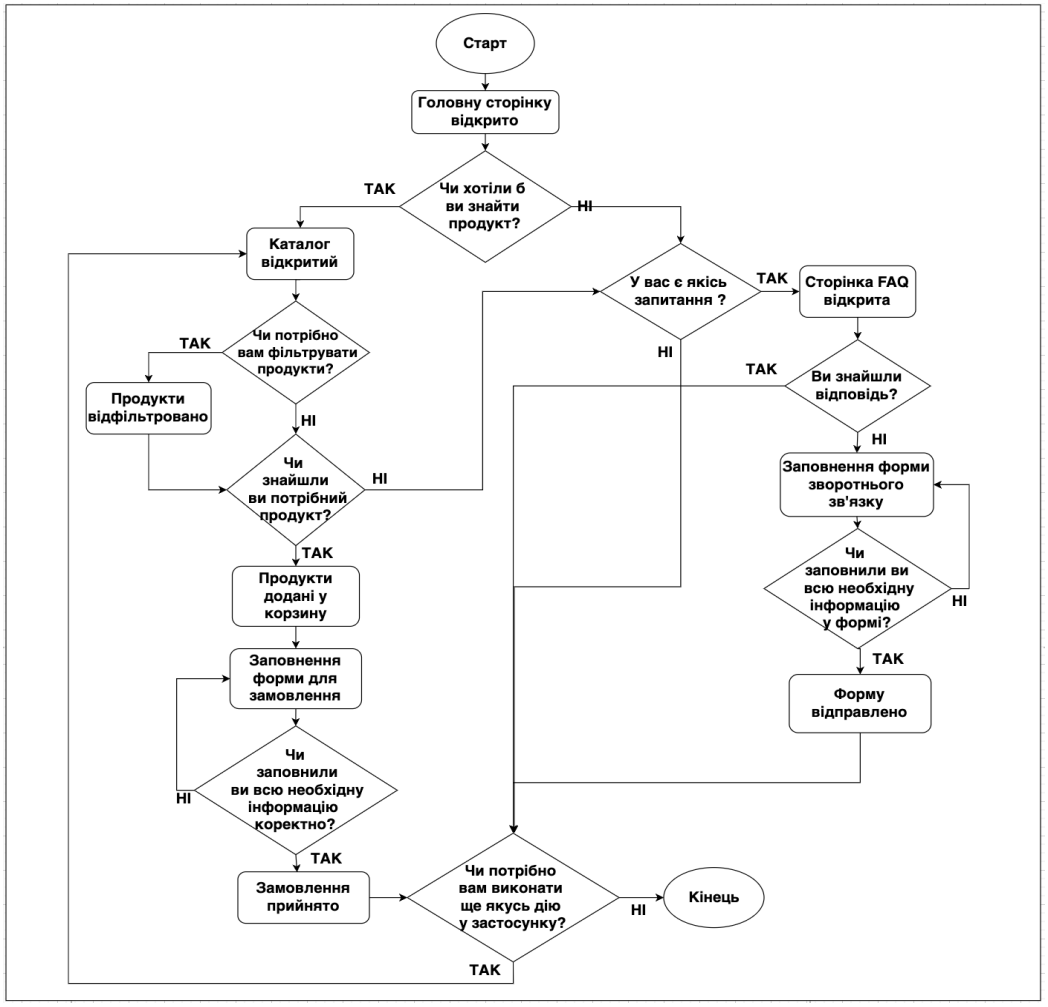


Рис. 3.7 Діаграма шляху користувача магазину Flower Lover

Використання UI Kit може допомогти створити красивий, функціональний та user-friendly веб-сайт.

UI kit (User Interface Kit) - це набір компонентів інтерфейсу користувача, які можна використовувати для створення послідовного та візуально привабливого дизайну веб-сайту. Використання UI kit може значно прискорити процес розробки та забезпечити високу якість дизайну.

Процес формування UI kit у Figma для веб-сайту з продажу квітів:

- Проведення дослідження веб-сайтів конкурентів та інших веб-сайтів з продажу квітів, щоб зрозуміти кращі практики дизайну та візуальні тренди.
- Створити карту користувацького шляху, щоб визначити ключові етапи, через які користувачі проходитимуть на веб-сайті.

- Формування списку компонентів, які вам знадобляться, наприклад, кнопки, поля введення, заголовки, зображення, картки продукту тощо.

Рис. 3.7 UI KIT у Figma для застосунку Flower Lover

Розроблений UI KIT став фундаментом для створення екранів магазину квітів Flower Lover у Figma. Завдяки йому вдалося зберегти візуальну послідовність. Усі компоненти інтерфейсу користувача на екранах мають однаковий стиль та оформлення, що робить дизайн гармонійним та привабливим.

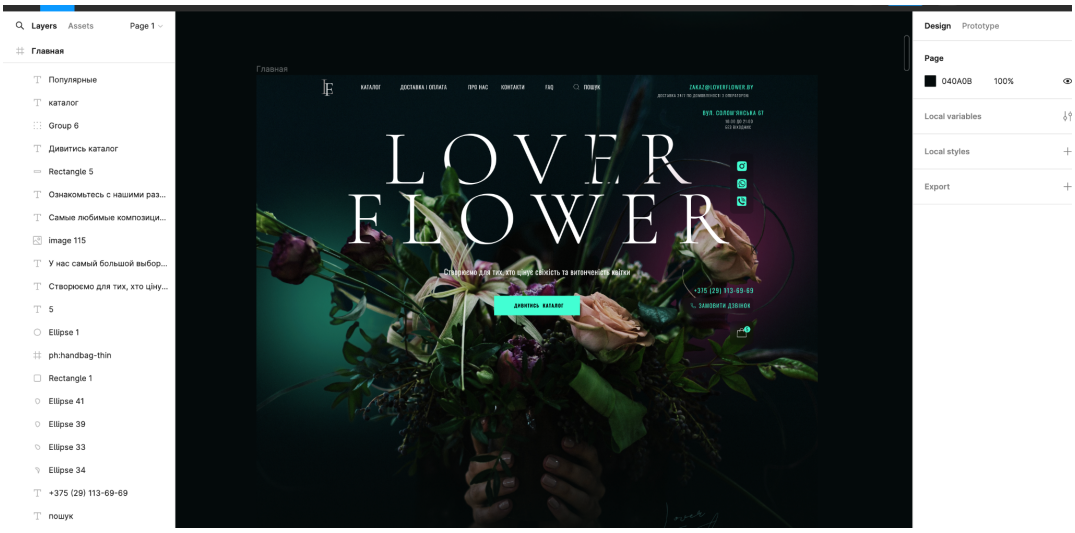


Рис. 3.8 Макет головної сторінки FlowerLover

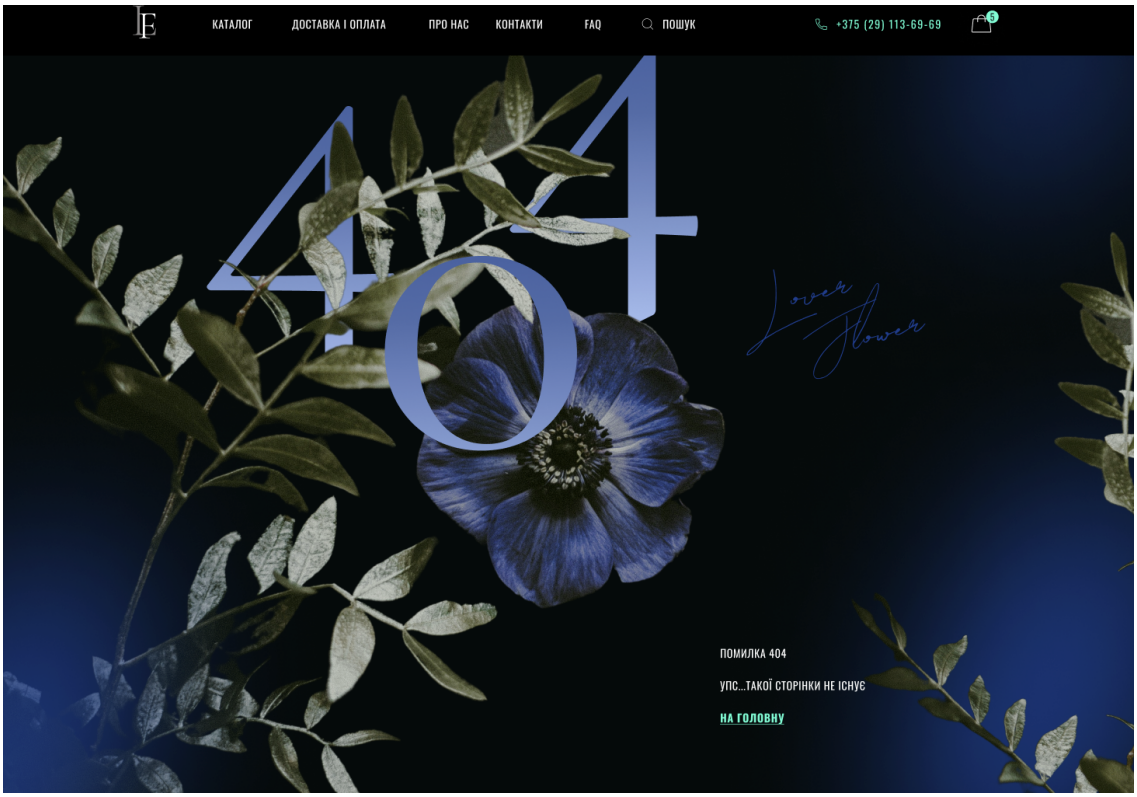


Рис. 3.9 Макет сторінки помилки 404

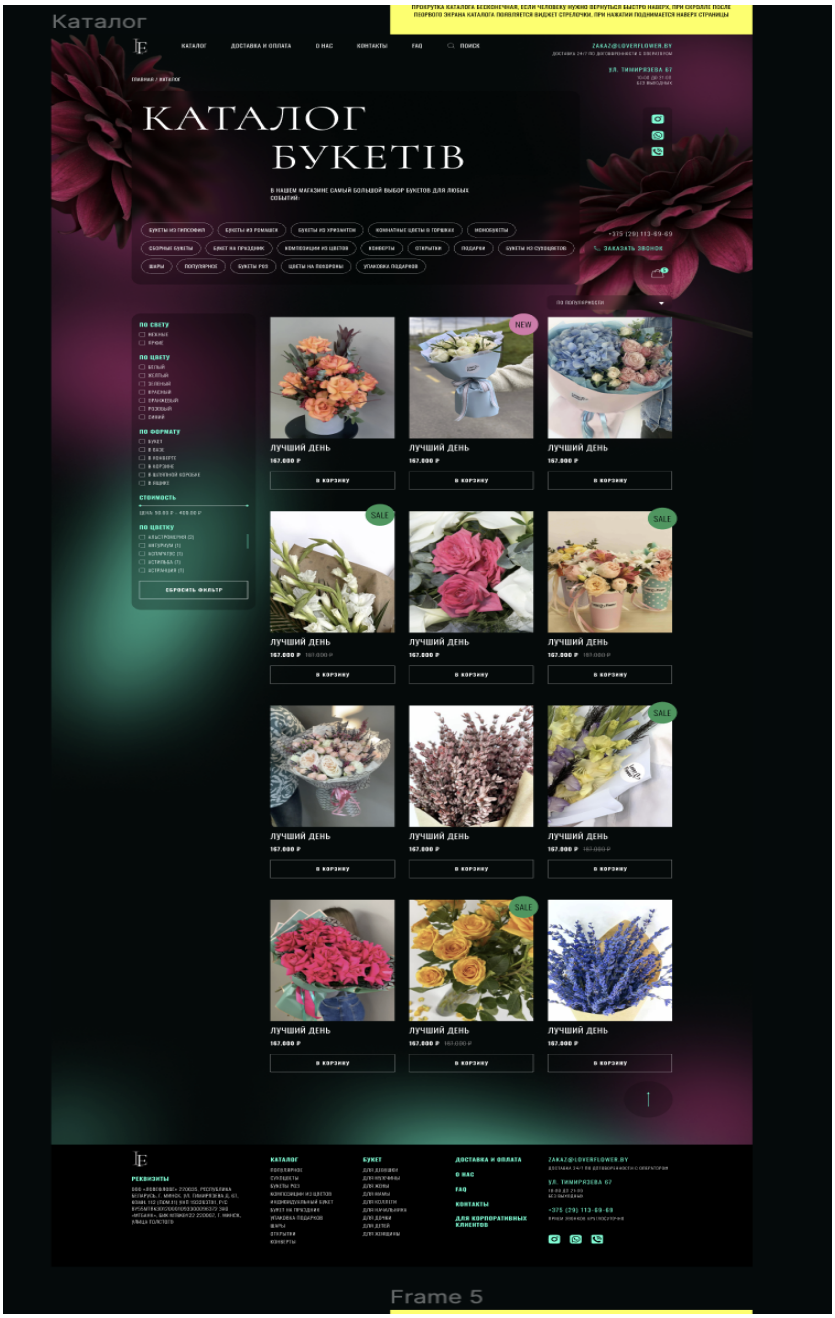


Рис. 3.10 Макет каталогу товарів

Завдяки UI kit вдалося створити чітко продумані та візуально привабливі екрани для магазину Flower Lover, які відповідають кращим практикам дизайну та забезпечують позитивний користувацький досвід.

3.7 Екрани веб-застосунку Flower Lover

Головний екран веб-застосунку для магазину квітів відіграє важливу роль у залученні та утриманні клієнтів, тому його дизайн та функціональність мають бути ретельно продумані.

Ключовими елементами головного екрану Flower Lover було обрано:

Великий банер з логотипом сайту. Цей банер слугує візитною карткою магазину квітів. Він має бути візуально привабливим, чітко відображати логотип та назву магазину, а також сформувати позитивне перше враження у користувачів.

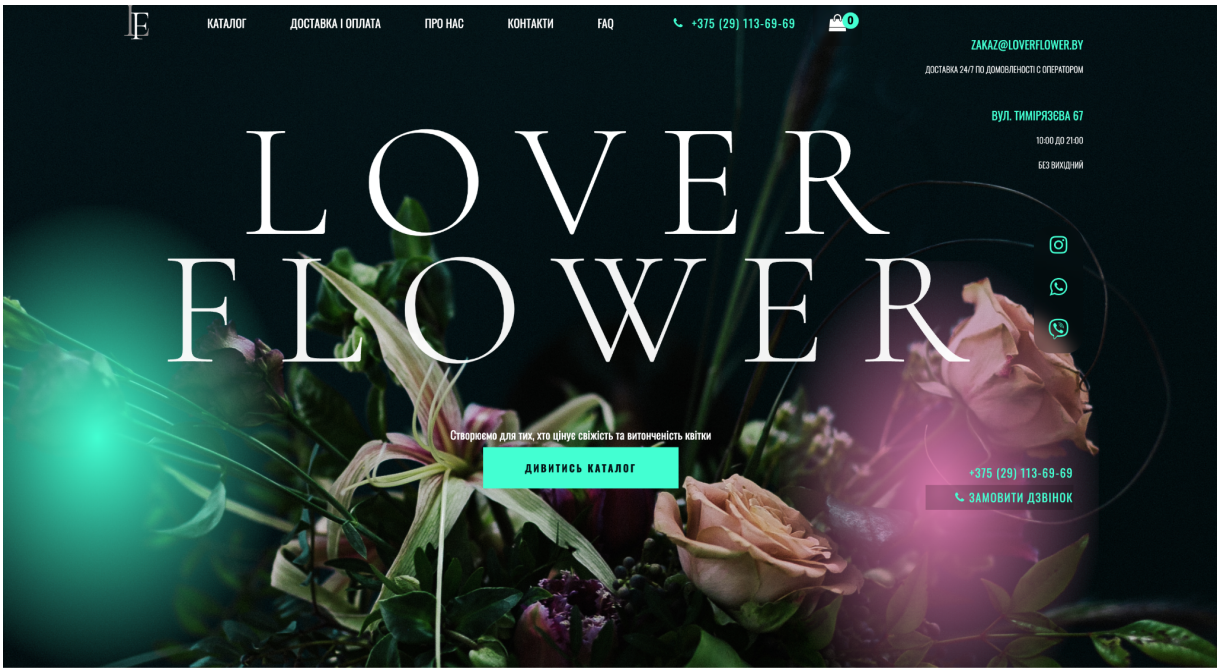


Рис. 3.11 Банер з логотипом на головній сторінці

Опис як зробити замовлення, цей блок має чітко та лаконічно пояснювати процес замовлення квітів, ключаючи кроки, які потрібно виконати, та доступні способи оплати та доставки.

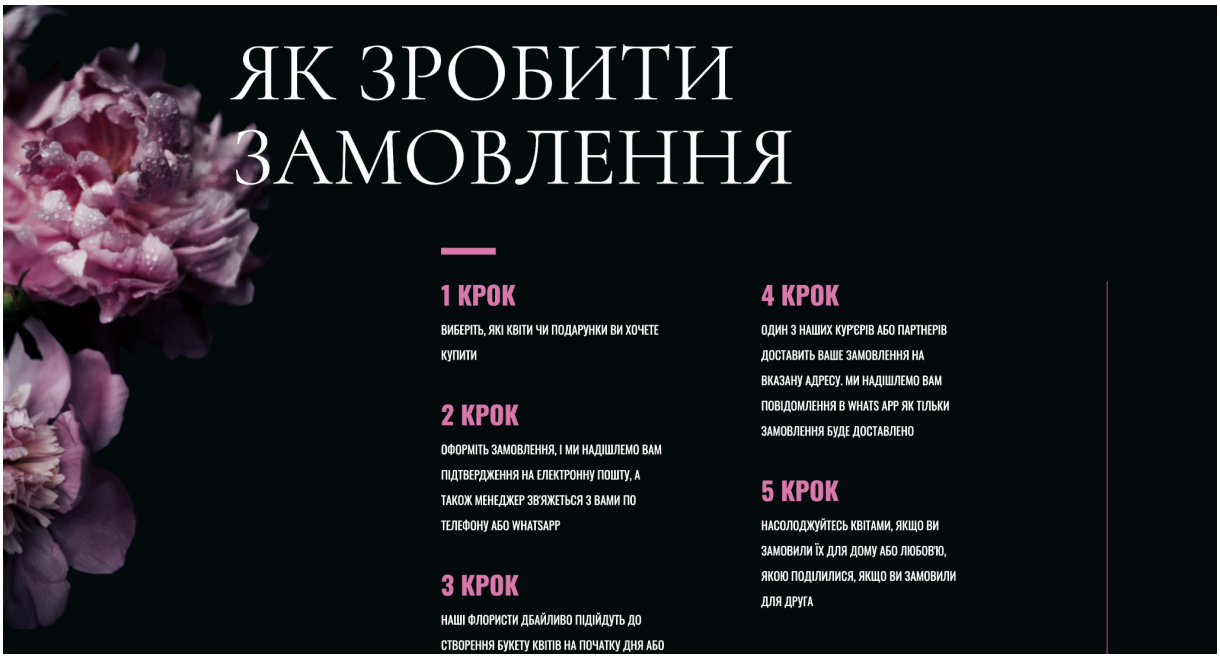


Рис. 3.12 Опис як зробити замовлення на головній сторінці

Слайдер з популярними товарами. Цей слайдер має демонструвати найпопулярніші квіткові композиції, новинки або сезонні пропозиції. Це допоможе користувачам швидко знайти те, що їм цікаво, та зацікавити їх у покупці.

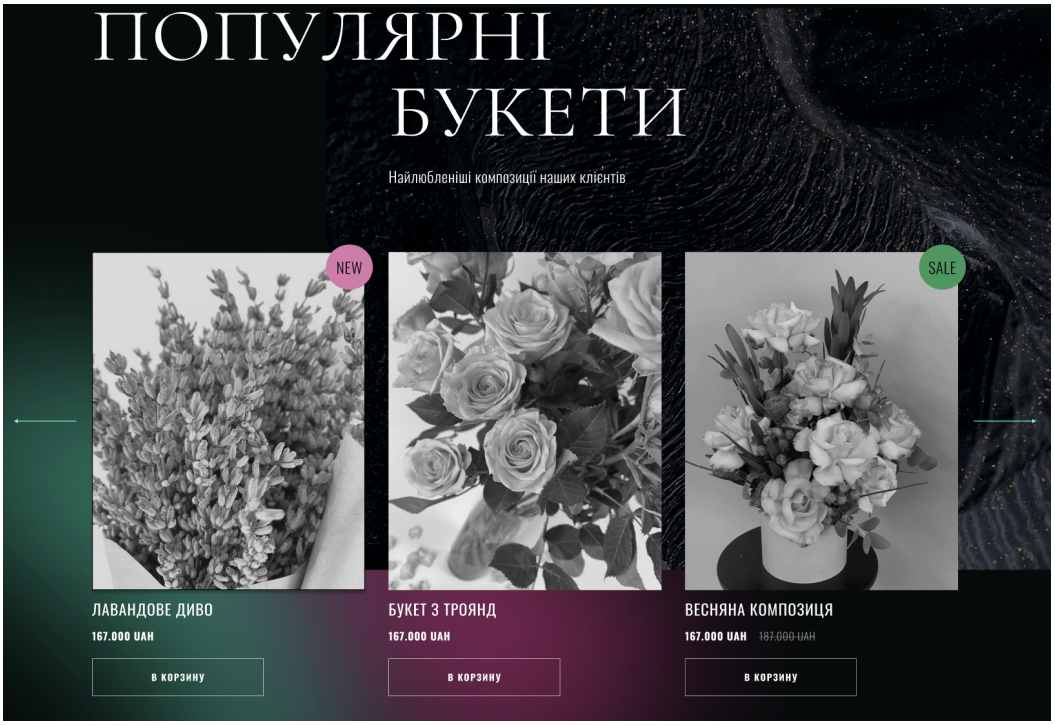


Рис. 3.13 Слайдер з популярними букетами на головній сторінці

Форма для зворотного зв'язку. Ця форма дозволяє користувачам залишати відгуки, задавати питання або зв'язатися з магазином. Це важливий канал комунікації, який допомагає магазину збирати зворотний зв'язок, покращувати обслуговування клієнтів та будувати лояльність.

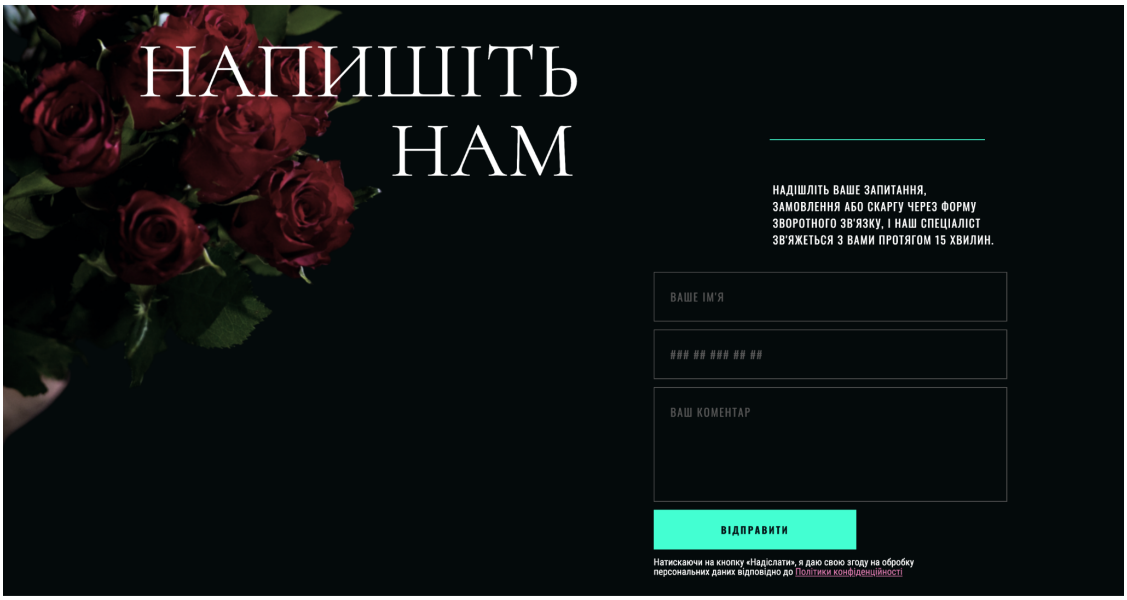


Рис. 3.14 Форма зворотнього зв’язку

Контактні дані, цей блок має містити всю необхідну контактну інформацію, включаючи адресу магазину, номер телефону, електронну адресу та години роботи.

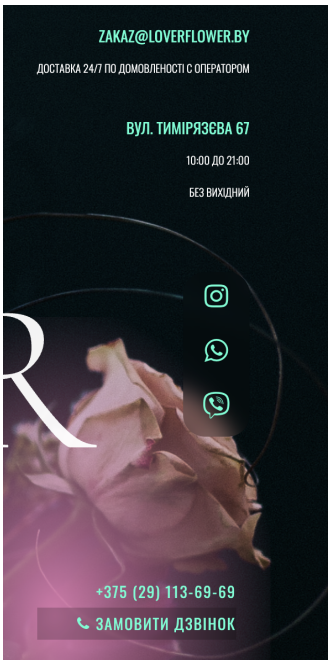


Рис. 3.15 Контактні дані на головні сторінці

Посилання на соціальні мережі дозволяють користувачам перейти на сторінки магазину в соціальних мережах, де вони можуть дізнатися більше про магазин, його продукцію та акції.

Рис. 3.16 Посилання на соціальні мережі на головні сторінці

Головний екран веб-застосунку для магазину квітів має бути інформативним, візуально привабливим та зручним для користувачів. Ретельно обдумуючи дизайн та функціональність головного екрану, можна зробити веб-застосунок більш ефективним інструментом для залучення та утримання клієнтів, а також для збільшення продажів квітів.

4 ТЕСТУВАННЯ ВЕБ-ЗАСТОСУНКУ

4.1 Опис підходу до тестування застосунку

Мануальне тестування веб-сайтів є важливим етапом в процесі розробки, оскільки дозволяє виявити помилки та недоліки, які можуть вплинути на користувацький досвід. Під час цього процесу тестувальники ретельно перевіряють кожну функцію та інтерфейсний елемент сайту

Перед початком процесу, спеціалісти повинні зрозуміти вимоги до сайту та його очікувані результати. Після цього створюється план тестування, у якому визначаються обсяг тестування, терміни виконання та інші деталі.

Далі розробляються докладні тестові сценарії, які описують послідовність дій, які будуть виконувати тестувальники на сайті.

Під час виконання тестів спеціалісти перевіряють різні аспекти, такі як:

- правильність реалізації функцій;
- коректність відображення інтерфейсу;
- взаємодію з користувачем та інші;

Якщо виявляються помилки або недоліки, вони реєструються для подальшого виправлення. Після виправлення дефектів виконується повторне тестування для перевірки, чи були виправлені проблеми та чи не виникло нових дефектів. Також проводиться тестування сумісності з різними пристроями, операційними системами і браузерами, а також перевіряється зручність використання та безпека сайту. На завершення створюється детальний звіт, в якому фіксуються всі виявлені дефекти, проблеми та рекомендації для подальшого вдосконалення веб-сайту.

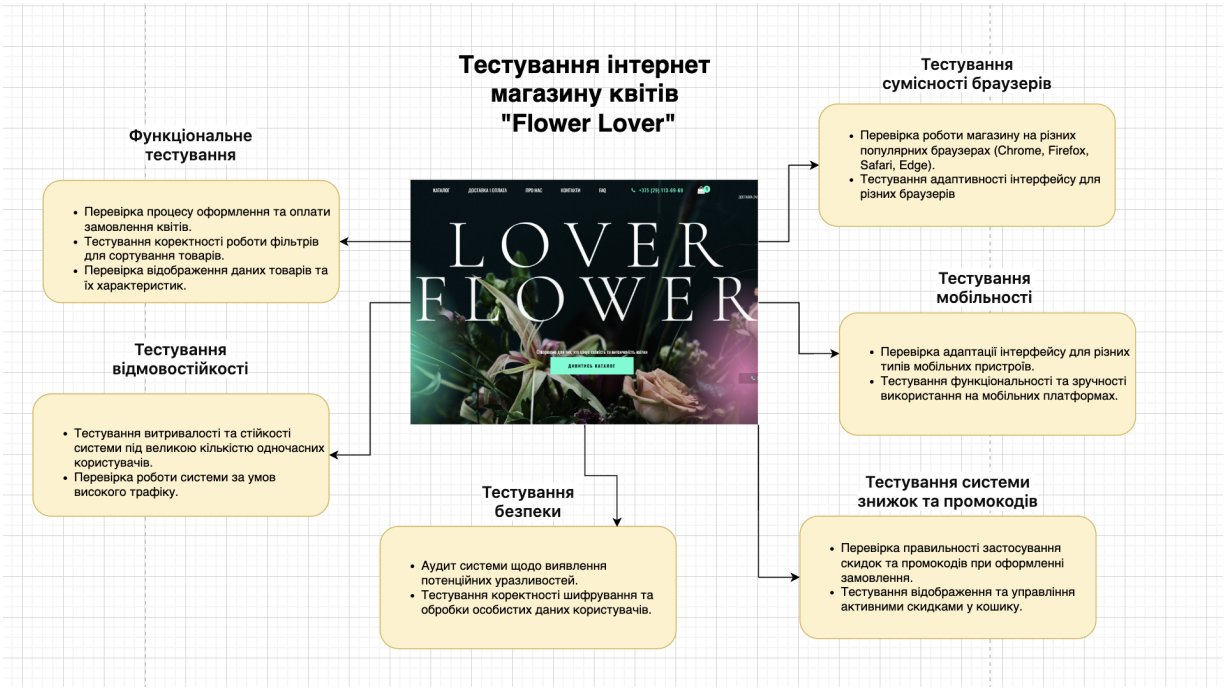


Рис. 4.1 Різновиди тестування веб-застосунку

Тест-плани необхідні для організації та проведення ефективного тестування кожного застосунку. Вони розробляються з урахуванням специфіки проекту та вимог з метою забезпечення якості продукту. У тест-планах описуються всі активності, пов'язані з тестуванням, з посиланням на відповідні політики, такі як тестування безпеки та навантажувальне тестування.

Структура тест-плану включає такі розділи:

1. Сфера застосування та загальне бачення. Опис обсягу тестування, включаючи функціонал, який підлягає тестуванню, та потреби клієнта, які задовольняються цим функціоналом.

2. Рівні тестування. Опис рівнів тестування, виконавців, цілей тестування та розкладу його проведення.

3. Види тестування. Опис використовуваних видів тестування та їх цілей, включаючи особливості, які можуть бути.

4. Ролі та відповідальність. Визначення ролей учасників проекту та їх зон відповідальності.

5. Вимоги до оточення. Опис необхідного середовища для проведення тестування, включаючи інтеграції та інструменти.

6. Тестові інструменти. Перелік використовуваних інструментів та їх призначення.

7. Результати тестування. Перелік артефактів тестування, які створюються та використовуються під час тестування.

8. Метрики тестування. Опис метрик тестування та їх збір, аналіз та використання.

9. Матриця покриття. Відображення покриття тест-кейсами вимог та загальне покриття.

10. Інструменти для репортингу. Опис правил оформлення багів та звітів, які використовуються під час тестування.

11. Загальний тестовий звіт. Вказівка видів тестування, які включаються у звіт, його частота та аудиторія.

12. Опис процесу розробки та тестування. Визначення фаз SDLC, процесу STLC, учасників, видів тестування та критерії входу/виходу.

13. Вимоги. Опис роботи з вимогами, включаючи функціональні та нефункціональні вимоги.

Такий тест-план допоможе забезпечити ефективне тестування, адаптоване під потреби та особливості кожного проекту.

4.2 Покриття застосунку тест-кейсами

Ретельне тестування допомагає виявити та виправити помилки, а також гарантувати, що застосунок відповідає очікуванням користувачів. Для тестування веб-застосунку для продажу квітів буде використовуватися метод чорної скриньки. Цей метод передбачає тестування функціональності застосунку без знання його внутрішньої структури та коду.

Перед тим, як розпочинати формування тест-кейсів, важливо чітко визначити мету тестування. У цьому випадку метою буде перевірка функціональності веб-застосунку для продажу квітів, включаючи фільтрацію товарів, відкриття картки товару, відправку форми зворотнього зв'язку, додавання товарів у корзину та оформлення замовлення.

Для зручного оформлення тест-кейсів я буду використовувати таблицю з наступними стовпчиками:

ID тесту: Унікальний номер для кожного тесту.

Опис тесту: Детальний опис того, що буде тестуватися.

Очікуваний результат: Очікувана поведінка веб-застосунку.

Фактичний результат: Зафіксований результат виконання тесту.

Статус: "Пройдено", "Не пройдено", "Блоковано" або "Неактуально".

Примітки: Додаткова інформація щодо тесту, наприклад, про причину невдачі.

Після створення таблиці тест-кейсів було виконано всі тести та зафіксувано результати в стовпчику "Фактичний результат". Після завершення тестування важливо проаналізувати результати та визначити, чи всі тести були успішно пройдені. Якщо деякі тести не пройшли, необхідно дослідити причини невдач та вжити заходів щодо їх виправлення. У випадку тестування Flower Lover всі тести було пройдено успішно.

Елементи тест-репорту після пройденого тестування:

ID тесту: 001

Перевірити фільтрацію товарів за категорією “По кольору” із вибраним параметром "Білий"

Опис тесту: Перевірити, чи можна фільтрувати товари за категорією “По кольору” із вибраним параметром "Білий" та чи відображаються лише товари, що належать до цієї категорії.

Кроки виконання:

1. Перейти на сторінку з каталог товарів.
2. У розділі "Фільтри" вибрати категорію за категорією “По кольору” із вибраним параметром "Білий"
3. Переконатися, що на сторінці відображаються лише товари, що належать до параметру "Білий"

Очікуваний результат:

На сторінці мають відображатися лише товари, що належать до категорії білі квіти.

Фактичний результат:

На сторінці відображають лише товари, що належать до категорії білі квіти.

Статус: пройдено успішно

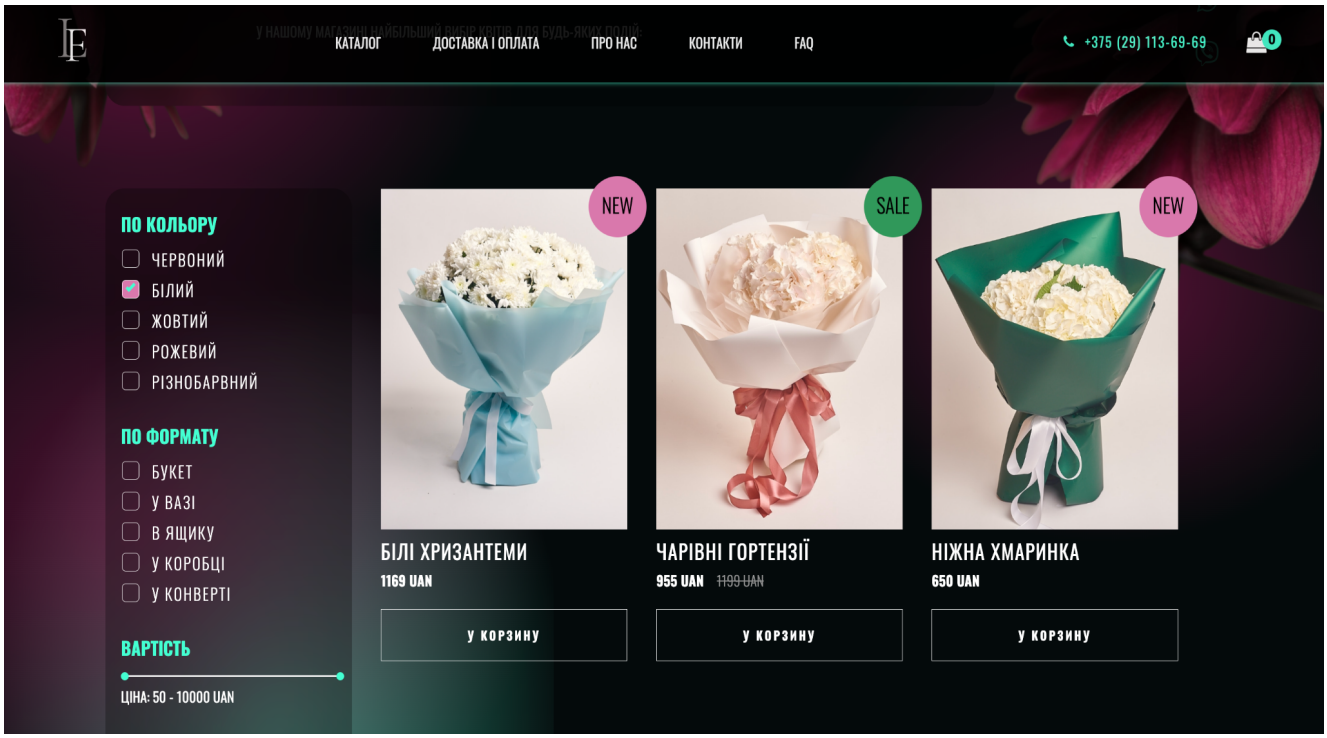


Рис. 4.1 Результат проходження тест-кейсу з ID 001

ID тесту: 002

Відкрити картку товару

Опис тесту: Перевірити, чи можна відкрити картку товару при натисканні на

зображення або назву товару, та чи відображається детальна інформація.

Кроки виконання:

1. Перейти на сторінку з товарами.
2. Обрати перший з товарів.
3. Натиснути на зображення або назву товару .
4. Переконатися, що відкрилася картка відповідного товару .
5. Переконатися, що на картці товару відображається детальна інформація, включаючи опис, фотографії, ціну та доступні варіанти.

Очікуваний результат:

Має відкритися картка товару з детальною інформацією.

Фактичний результат:

Відкрилася картка товару з детальною інформацією.

Статус: пройдено успішно

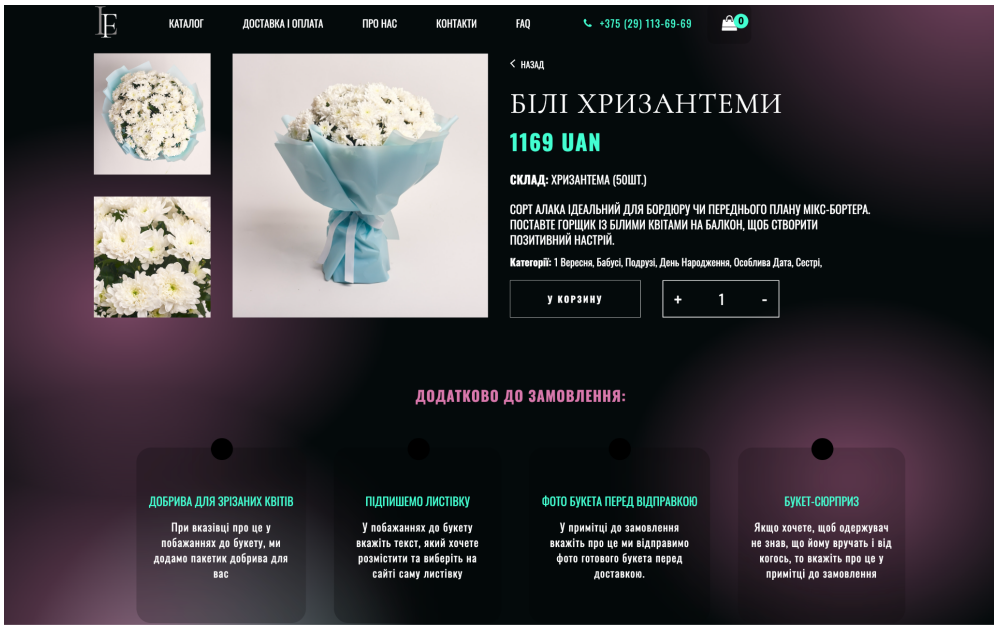


Рис. 4.2 Результат проходження тест-кейсу з ID 002

ID тесту: 007

Відправка форми зворотнього зв'язку з незаповненими полями

Опис тесту: Перевірити, чи неможливо відправити форму зворотнього зв'язку з незаповненими полями "Ім'я" та "Номер телефону", а також чи підсвічуються

незаповнені поля червоним при спробі відправлення.

Кроки виконання:

1. Перейти на сторінку з формою зворотнього зв'язку.
2. Натиснути кнопку "Відправити".

Очікуваний результат:

1. Форма не має бути відправлена.
2. Незаповнені поля "Ім'я" та "Номер телефону" мають світитися червоним.

Фактичний результат:

1. Форму не відправлено.
2. Незаповнені поля "Ім'я" та "Номер телефону" світяться червоним.

Статус: пройдено успішно.

Примітки:

1. Перевірити, чи з'являється повідомлення про помилку, що вказує на незаповнені поля.
2. Перевірити, чи підсвічуються червоним інші незаповнені поля, якщо такі є.

Додаткові кроки:

1. Спробувати відправити форму з одним із полів заповненим, а іншим - ні.
2. Спробувати ввести некоректні дані в поля (наприклад, невірний формат номера телефону).
3. Перевірити, чи з'являються відповідні повідомлення про помилку.

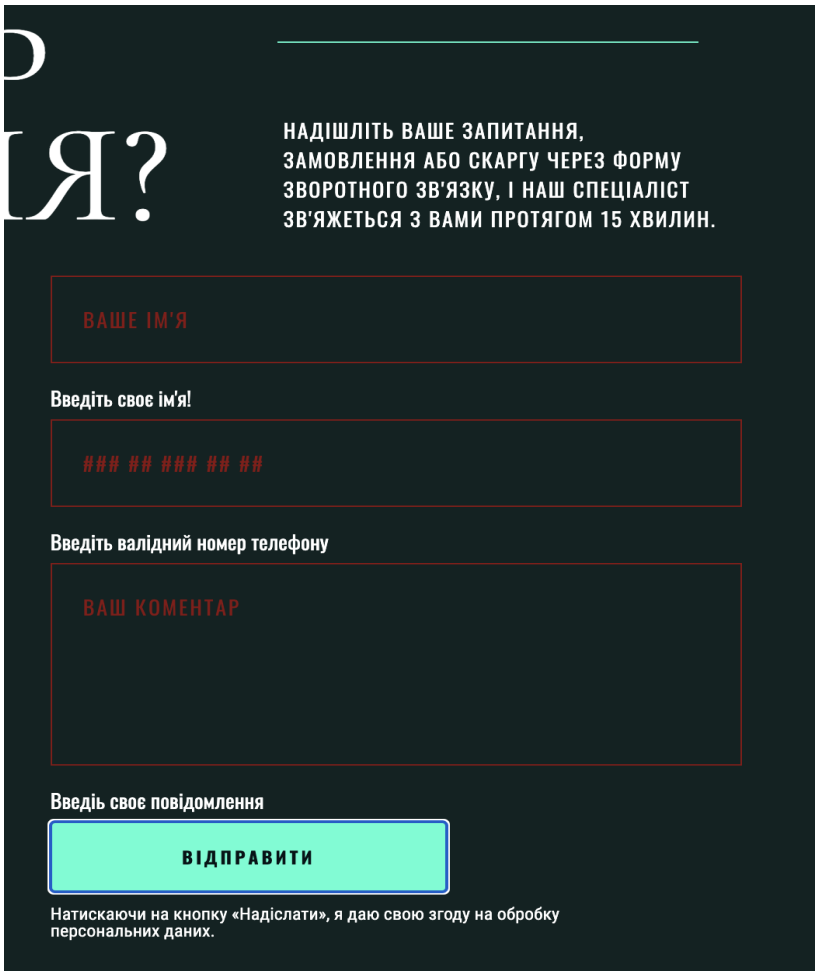


Рис. 4.2 Результат проходження тест-кейсу з ID 007

Тестування - це важливий процес, який допомагає створити якісний веб-застосунок. Завдяки ретельному тестуванню можна виявити та виправити помилки, а також гарантувати, що застосунок відповідає очікуванням користувачів.

ВИСНОВКИ

У результаті виконання дипломної роботи розроблено веб-застосунок для продажу квіткових композицій з використанням мови JavaScript та фреймворку Vue.js. Актуальність даної роботи полягає в тому, що наразі існує потреба у зручних та сучасних онлайн-інструментах для продажу квітів.

В ході виконання дипломної роботи було виконано наступні завдання:

1. Проведено аналіз предметної області та досліджено існуючі веб-сайти та онлайн-магазини з продажу квіткових композицій. Визначено їхні сильні та слабкі сторони, а також потреби та очікування цільової аудиторії.

2. Розроблено функціональні та нефункціональні вимоги до веб-застосунку. Визначено необхідний набір функцій, інтерфейс користувача, дизайн, а також характеристики продуктивності, безпеки та масштабованості.

3. Обґрунтовано вибір мови програмування JavaScript та фреймворку Vue.js для розробки веб-застосунку. Vue.js є сучасним, легким у вивченні та динамічно розвиваючимся фреймворком, який добре підходить для створення інтерактивних інтерфейсів користувача.

4. Спроектовано та розроблено веб-застосунок з використанням Vue.js. Було реалізовано основні функції, що покривають потреби користувачів, котрі мають намір здійснити покупку або отримати консультацію.

5. Проведено тестування веб-застосунку для виявлення та виправлення помилок. Перевірено виконання всіх функціональних та нефункціональних вимог. Тестування пройдено успішно.

6. Проект пройшов апробацію на Науково-технічних конференціях «Сучасні інтелектуальні інформаційні технології в науці та освіті» Київ, ДУІКТ, 18 травня 2024 року та «Застосування програмного забезпечення в ІКТ» Київ, ДУІКТ, 23 квітня 2024 року.

ПЕРЕЛІК ПОСИЛАНЬ

1. Гленфорд Майерс, Том Баджетт, Кори Сандлер. Мистецтво тестування програм, 3-тє видання: книга. Вид. 3-тє, переробл. і допов. Київ, 2018, 192 с.
2. Ерік Фрімен, Елізабет Робсон. Head First. Програмування на JavaScript: книга. Вид. 3-тє, переробл. і допов. Київ, 2022, 672 с.
3. Evan You. The Progressive JavaScript Framework 15.12.2023 URL: https://vuejs.org/guide/introduction.html (дата звернення 13.04.2024)
4. Фреймворк Node.js — ідеальний інструмент для створення мережевих застосунків. 10.10.2020 URL:https://gl.ua/pl/blog/freymvork-nodejs-idealnyy-instrument-dlya-stvorennya-merezhevykh (дата звернення 1.04.2024)
5. Ткаченко Аліна. Найкращі фреймворки Node.js: розбираємо їх особливості. 30.06.2023. URL: https://wezom.com.ua/ua/blog/luchshie-freymvorki-nodejs (дата звернення 29.03.2024)
6. Кори Сандлер, Том Банджетт. Мистецтво тестування програм: книга. Вид. 3-тє, допов. Київ, 2020 272 с.
7. Роберт Сесіл Мартін. Чиста архітектура 3-е видання: книга. Вид. 3-тє, переробл. і допов. Київ, 2019, 368 с.
8. Закон України "Про електронну комерцію" згідно із Законом № 1089-IX від 16.12.2020. URL: https://zakon.rada.gov.ua/laws/main/index (дата звернення: 15.05.2024).
9. Daniel Kelly. How to structure large scale Vue.js application. 06.07.2021. URL: https://vueschool.io/articles/vuejs-tutorials/how-to-structure-a-large-scale-vue-js-application/ (дата звернення 05.04.2024)
10. Viktor Misiko. Implementing Clean Architecture in a Vue.js application 09.05.2024. URL: https://medium.com/@victormisiko.vm/implementing-clean-architecture-in-a-vue-js-application-fd23b33ef488 (дата звернення 09.05.2024)
11. Еллен Лаптен, Дженніфер Коул. Графічний дизайн. Нові основи : книга. Вид. 1-е, допов. Київ, 2020 264с.
12. Rahul Gulati. A five steps to design an amazing web app. 08.09.2022 URL: https://www.hotjar.com/web-app-design/ (дата звернення: 15.04.2024)

13. Закон України "Про захист персональних даних" згідно із Законом № 5491-VI від 20.11.2012 URL: https://zakon.rada.gov.ua/rada/show/2297-17#Text (дата звернення: 10.05.2024).

14. Пілевич Д. С. Застосування системного підходу до розгляду сутності електронної комерції *Бізнес Інформ* 2019. №2. С. 109–114. URL: https://doi.org/10.32983/2222-4459-2019-2-109-114

15. Резнікова В. Поняття, значення та перспективи правового забезпечення електронної комерції в Україні. *Теорія і практика інтелектуальної власності*. 2015. № 2. С. 58–72. URL: http://nbuv.gov.ua/UJRN/Tpiv_2015_2_10

16. Joe Johnson. How to build a Web App. 01.24.2024. URL: https://budibase.com/blog/how-to-make-a-web-app/ (дата звернення: 19.04.2024)

17. Дорон Маєр. Workflow. Практичний посібник творчого процесу : книга. Вид. 1-е, допов. Київ, 2020 с.

18. Шкарлет С. М., Гонта О. І., Дубина М. В. Особливості застосування системного підходу до пізнання економічних явищ. *Науковий вісник Полісся*. 2016. № 4. Ч. 1. С. 9–17. URL: http://ppeu.stu.cn.ua/article/view/184123

19. Юлія Д.Р. Посібник з SEO для інтернет-магазину 01.12.2023 URL: https://hostiq.ua/blog/ukr/seo-ecommerce-guide/ (дата звернення 05.04.2024)

20. Анатолій Буравін. Прогресивний web-додаток та SEO 05.11.2019 URL: https://seo-studio.ua/blog/web-prolojeniye-seo (дата звернення 16.04.2024)

21. Джош Сейден, Джефф Готельф. Lean UX: Створення класних продуктів із командами Agile 2-ге видання: книга. Вид. 2-ге, Київ, 2024, 206 с.

22. Нідзельська В.Р., Негоденко О.В. Вплив дизайну на успішність веб-застосунку для продажу квіткових композицій. *IV Всеукраїнської Науково-практична конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті»*, 15 травня 2024 р., Київ, Державний університет інформаційно-телекомунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С.212.

23. Нідзельська В.Р., Негоденко О.В. Визначення вимог для веб-застосунку для продажу квіткових композицій. *Всеукраїнської науково-технічна конференції «Застосування програмного забезпечення в ІКТ»*. 24 квітня 2024 р., Київ, Державний університет інформаційно-телекомунікаційних технологій. Збірник тез. К.: ДУІКТ, 2024. С.414.

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛ
(Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка web-застосунку для продажу квіткових композицій мовою JavaScript з використанням фреймворку Vue.js

Виконала студентка 4 курсу
групи ПД-41
Нідзельська Вікторія Русланівна
Керівник роботи
К.т.н., доц., доцент кафедри ІПЗ Негоденко Олена Василівна

Київ – 2024

1

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** - автоматизація бізнес-процесів у сфері продажу квіткових композицій шляхом застосування веб-застосунку, створеного мовою Javascript з використанням фреймворку Vue.js.
- **Об’єкт дослідження** - процес продажу квіткових композицій в онлайн магазині.
- **Предмет дослідження** – програмне забезпечення для продажу квіткових композицій.

2

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Дослідити ринок квіткової індустрії та ідентифікувати основних конкурентів.
2. Оцінити їхні веб-платформи та функціонал для розуміння сильних і слабких сторін.
3. Сформулювати вимоги до веб-застосунку, враховуючи функціональність та зручність використання.
4. Проаналізувати можливості використання різних технологій та фреймворків для розробки веб-застосунку.
5. Спроектувати та розробити веб-застосунок для продажу квіткових композицій.
6. Провести тестування сумісності та функціональне для перевірки відповідності вимогам.

3

АНАЛІЗ АНАЛОГІВ

Функціональні та нефункціональні характеристики	**Dicentra**	**Don Pion**	**Камелія**	**Lepestki**	**Flower Lover**
Каталог букетів	+	+	+	+	+
Фільтрація по ціні	+	-	-	+	+
Окрема сторінка товару	+	-	+	-	+
Форма зворотнього зв'язку	+	+	-	-	+
Адаптивність під різні розширення екрану	+	-	+	-	+
Середня оцінка дизайну інтерфейсу від користувачів	3,8	3	2,9	1,4	4,9
Швидкість завантаження елементів	2-5 с	5-8 с	2-5 с	5-7 с	<2 с

4

ВИМОГИ ДО ДОДАТКУ

Функціональні:

- Користувачі повинні мати можливість фільтрувати букети за ціною та типом.
- Наявність картки з коротким описом товару.
- Відкриття окремої сторінки з детальною інформацією про товар, при натисканні на картку товару.
- Користувачі повинні мати можливість додавати у кошик букети і видаляти їх.
- Кошик покупок повинен відображати назву, зображення, кількість та ціну кожного доданого букета.
- Користувачі повинні мати можливість ввести свою адресу доставки, контактні дані та вибрати спосіб оплати.
- Користувачі повинні мати можливість залишати відгуки про букети та послуги.

Нефункціональні:

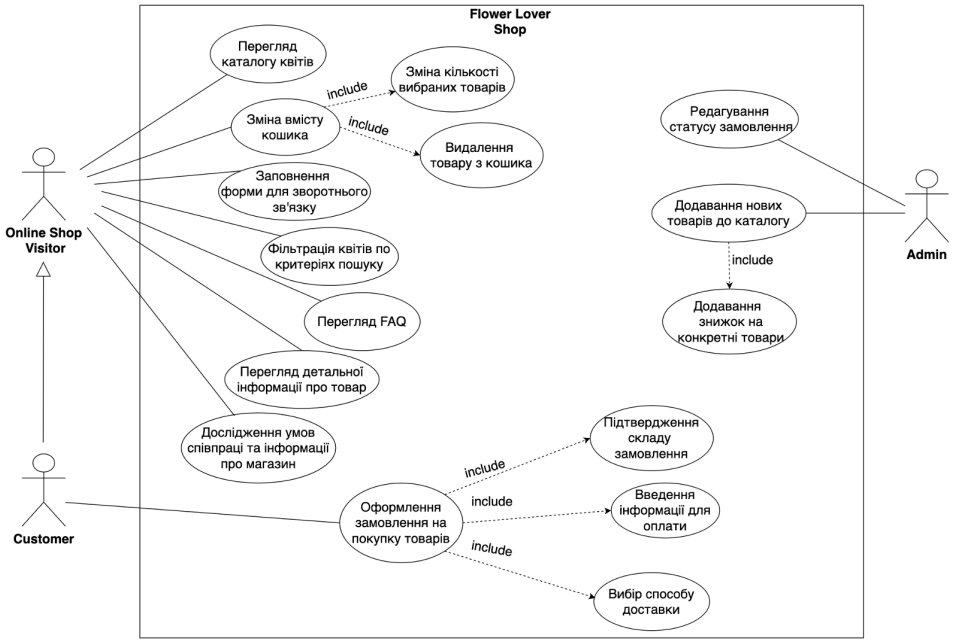
- Швидкість завантаження сторінок: 2 секунди або менше.
- Оптимізація контенту та структури сайту для покращення рейтингу в пошукових системах.
- Веб-додаток повинен бути адаптивним до різних пристроїв та екранів.
- Доступність для людей з вадами зору.
- Дизайн інтерфейсу повинен забезпечувати позитивний користувацький досвід.

5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

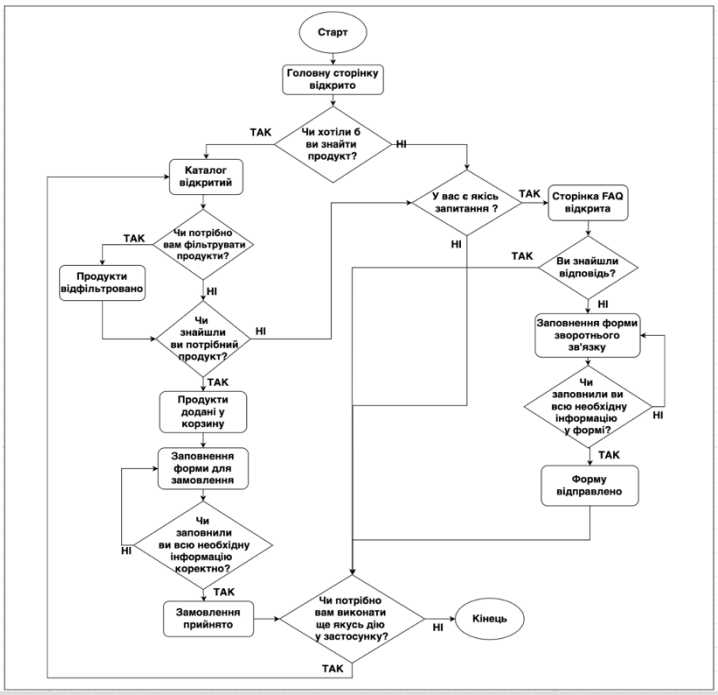
6

ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



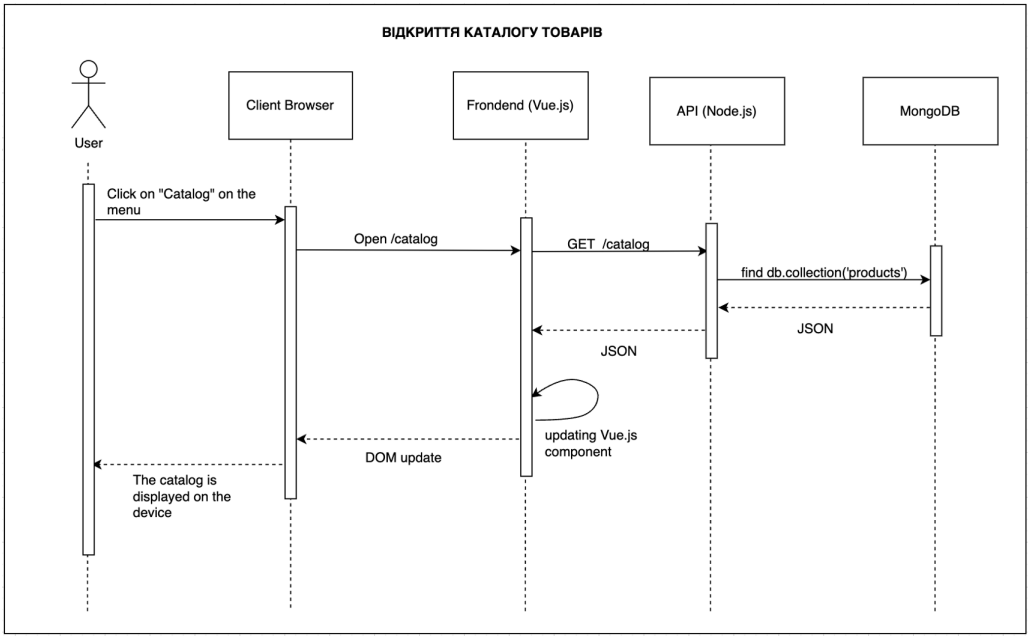
7

ДІАГРАМА ШЛЯХУ КОРИСТУВАЧА



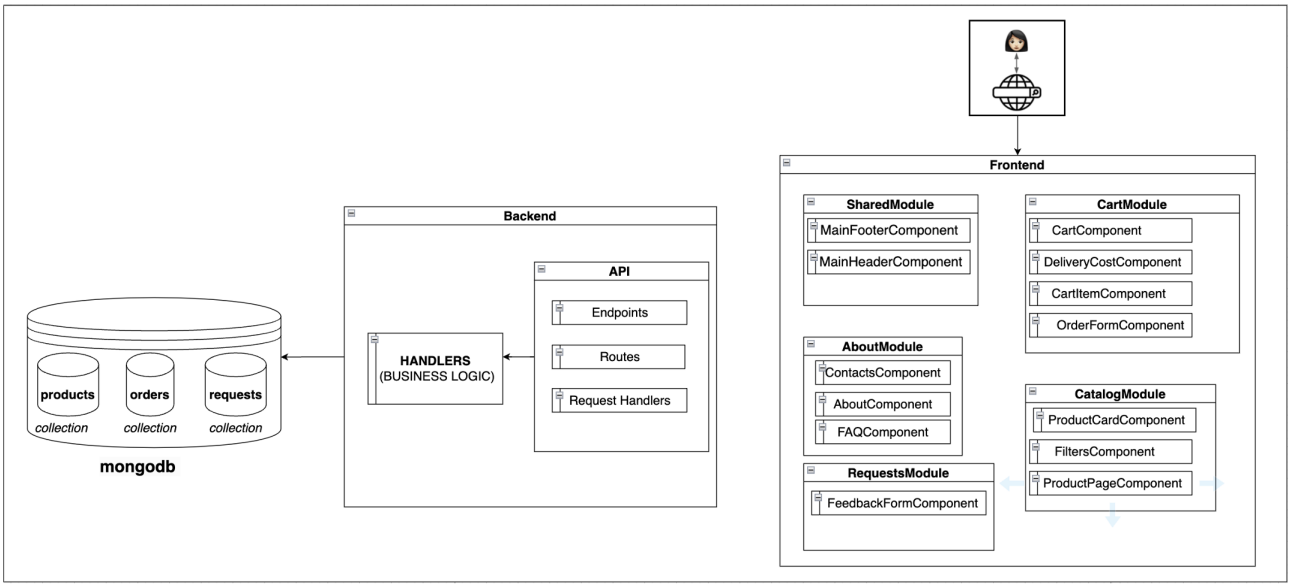
8

ДІАГРАМА ПОСЛІДОВНОСТІ



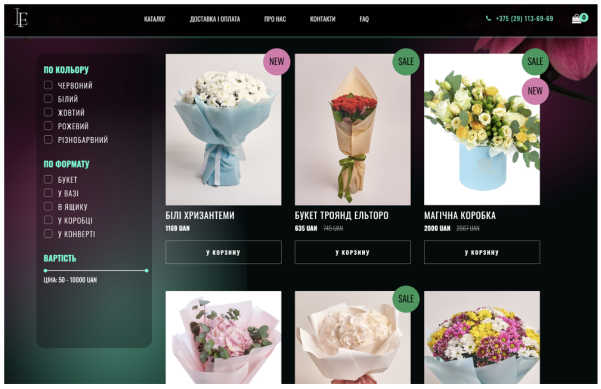
10

СХЕМА АРХІТЕКТУРИ ЗАСТОСУНКУ

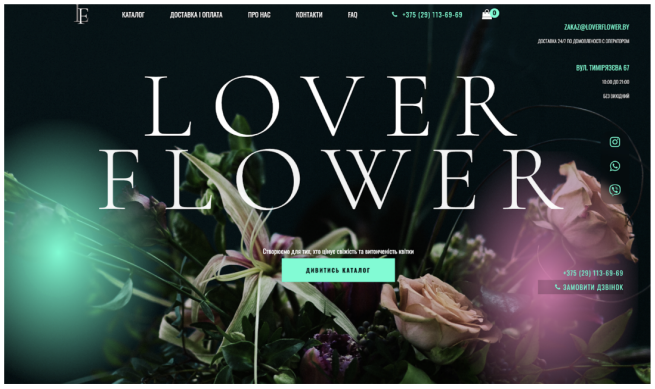


9

ЕКРАННІ ФОРМИ

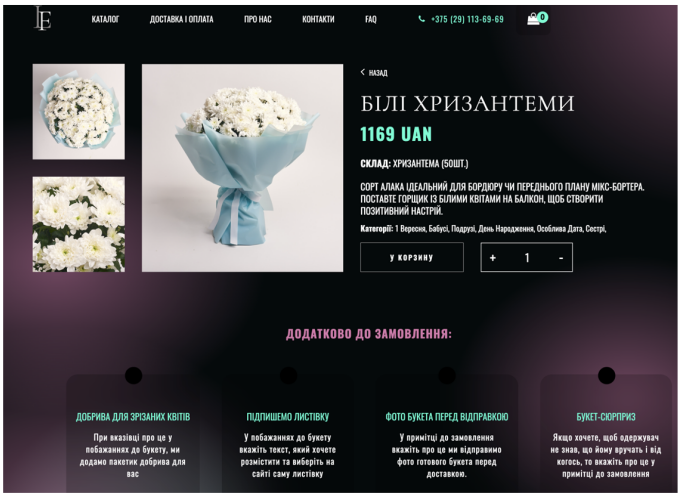


Каталог товарів

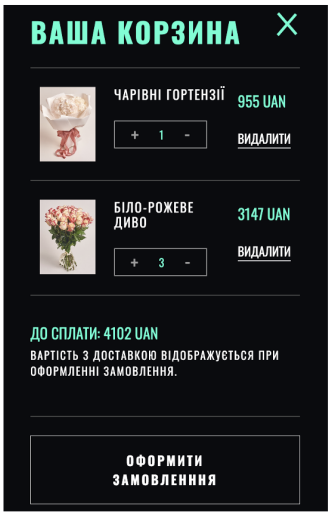


Головний екран

ЕКРАННІ ФОРМИ

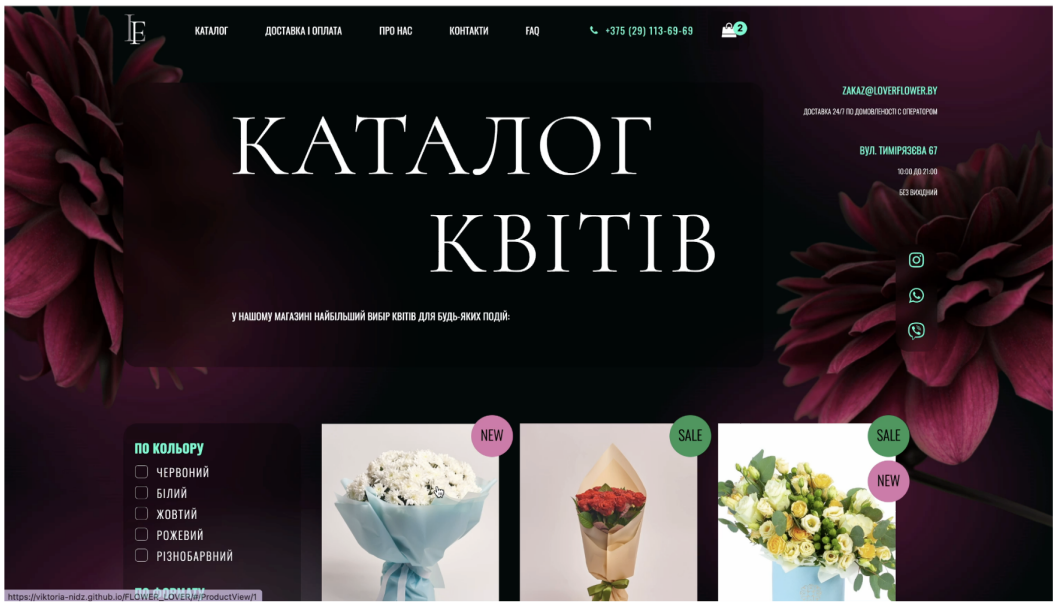


Екран товару



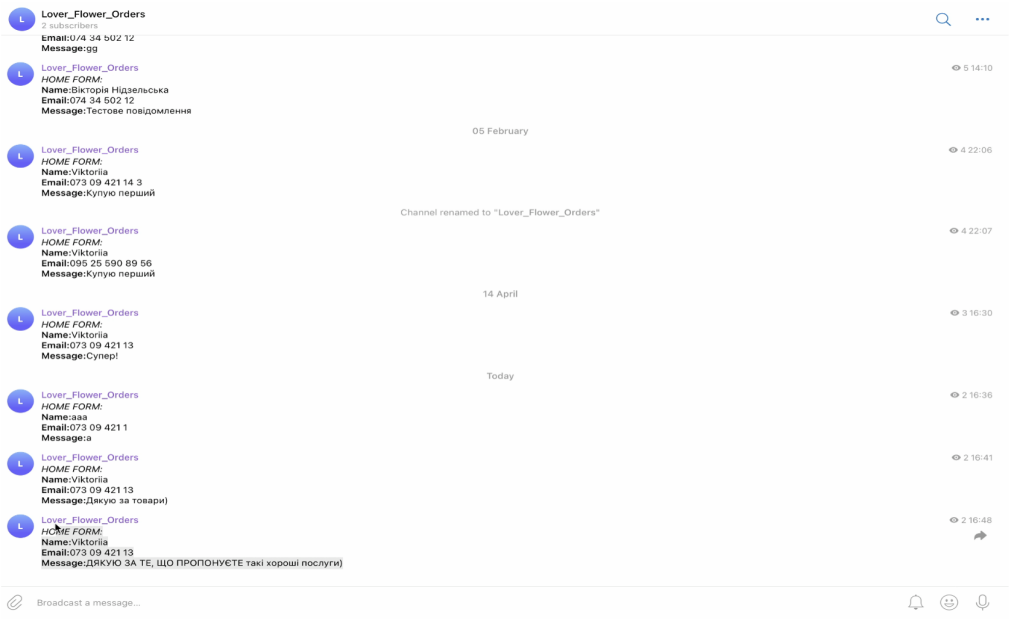
Корзина

Демонстрація роботи застосунку



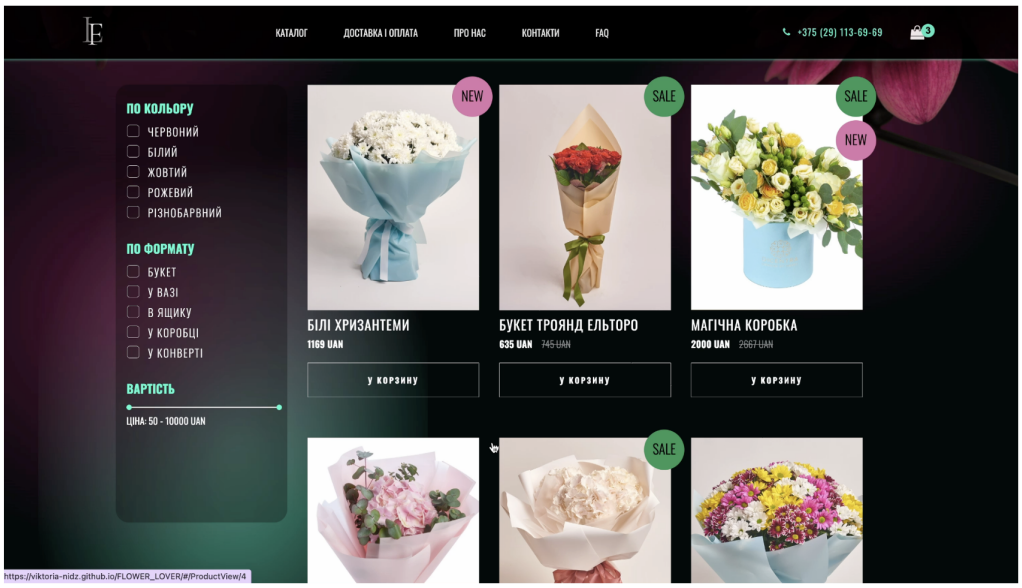
13

Демонстрація роботи застосунку



14

Демонстрація роботи застосунку



15

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Нідзельська В.Р., Негоденко О.В. Вплив дизайну на успішність веб-додатку для продажу квіткових композицій: матеріали IV Всеукраїнської Науково-практична конференції «Сучасні інтелектуальні інформаційні технології в науці та освіті» Подано до друку.

2. Нідзельська В.Р., Негоденко О.В. Визначення вимог для веб-додатку для продажу квіткових композицій: матеріали Всеукраїнської науково-технічна конференції «Застосування програмного забезпечення в ІКТ». Подано до друку.

16

ВИСНОВКИ

1. Проведено аналіз предметної області та досліджено існуючі веб-сайти та онлайн-магазини з продажу квіткових композицій. Визначено їхні сильні та слабкі сторони, а також потреби та очікування цільової аудиторії.
2. Розроблено функціональні та нефункціональні вимоги до веб-додатку. Визначено необхідний набір функцій, інтерфейс користувача, дизайн, а також характеристики продуктивності, безпеки та масштабованості.
3. Обґрунтовано вибір мови програмування JavaScript та фреймворку Vue.js для розробки веб-додатку.
4. Спроектовано та розроблено веб-додаток з використанням Vue.js. Було реалізовано основні функції, що покривають потреби користувачів, котрі мають намір здійснити покупку або отримати консультацію.
5. Проведено тестування веб-додатку для виявлення та виправлення помилок. Перевірено виконання всіх функціональних та нефункціональних вимог. При тестуванні помилок та відхилення від вимог не виявлено.

17

ДЯКУЮ ЗА УВАГУ!

18

ДОДАТОК Б. ЛІСТИНГИ ОСНОВНИХ МОДУЛІВ

```
<template>
 <div class="faq_blok">
  <div class="question_wrap" @click="toggleFaq()">
   <div class="block_wrap">
    <div class="question">{{ faq.question }}</div>
    <button type="button" v-if="isOpen">-</button>
    <button type="button" v-else>+</button>
   </div>

   <div v-if="isOpen" class="answer">{{ faq.answer }}</div>
  </div>
 </div>
</template>

<script>
export default {
 name: "FAQ",
 props: ["faq", "i"],

 data() {
  return {
   isOpen: false,
  };
 },

 methods: {
  toggleFaq() {
   if (this.isOpen === false) {
    this.isOpen = true;
   } else {
    this.isOpen = false;
   }
  },
 },
};
</script>

<style lang="scss" scoped>
$lightGreeen-color: #43ffd2;
$pastelPinc-color: #d978ac;
$pastelPinc-colorSpot: #d978ac8b;
$lightGreeen-colorSpot: #43ffd399;

// FAQ PAGE START
.block_wrap {
 display: flex;
 justify-content: space-between;
}
.faq {
 .faq_block {
  padding-bottom: 200px;
 }
 .question {
  max-width: 825px;
  margin-bottom: 15px;
  font-weight: 400;
  color: $lightGreeen-color;
  text-transform: uppercase;
```

```
    font-size: 20px;
  }
  .question_wrap {
    padding: 20px;
    margin-bottom: 10px;
    border: 1px solid $lightGreeen-color;
  }

  .answer {
    max-width: 825px;
    font-size: 18px;
    line-height: 29px;
    font-weight: 300;
    font-family: "Oswald";
    letter-spacing: 1px;
  }
}
// FAQ PAGE END
</style>
<template>
  <div class="cart_item" v-for="flower in cart" :key="flower">
    <div class="center_block">
      <img
        :src="require(`@/assets/images/flowers/${flower.id}.1.jpg`)"
        class="card-img-top"
        :alt="flower.title"
      />

      <div class="wrap_qty_title">
        <div class="flower_title">
          {{ flower.name }}
        </div>
        <div class="qty_change">
          <button
            type="button"
            class="_inc"
            @click="changeProductQty(flower.id, 'inc')"
          >
            +
          </button>
          <div>{{ flower.qty }}</div>
          <button
            type="button"
            class="qty_dec"
            @click="changeProductQty(flower.id, 'dec')"
          >
            -
          </button>
        </div>
      </div>
    </div>
    <div class="total_block">
      <div class="price_item">{{ flower.total }} UAN</div>
      <button type="button" class="delete_btn" @click="askProdDel(flower.id)">
        видалити
      </button>
    </div>
  </div>
  <div class="cart_footer">
    <div class="to_pay">До сплати: {{ this.total }} UAN</div>
    <div class="warning">
```

```
      Вартість з доставкою відображується при оформленні замовлення.
    </div>
  </div>
</template>

<style lang="scss" scoped>
$lightGreeen-color: #43ffd2;
$pastelPinc-color: #d978ac;

.flower_title {
  color: #fff;
  text-transform: uppercase;
  font-size: 14px;
  letter-spacing: 1.2px;
  padding-bottom: 20px;
  max-width: 120px;
  line-height: 16px;
}
.product_qty {
  position: absolute;
  top: 14px;
  left: 36px;
  width: 20px;
  height: 20px;
  font-size: 14px;
  font-weight: 700;
  display: flex;
  justify-content: center;
  align-items: center;
  border-radius: 50%;
  background-color: $lightGreeen-color;
  color: black;
}

// CART BLOCK START
.cart_side {
  .to_pay {
    text-transform: uppercase;
    margin-top: 25px;
    font-size: 16px;
    color: $lightGreeen-color;
  }
  .warning {
    color: white;
    text-transform: uppercase;
    font-size: 12px;
    line-height: 17px;
    letter-spacing: 1.2px;
  }
  .delete_btn {
    background-color: transparent;
    border: none;
    color: white;
    border-bottom: 1px solid white;
    text-transform: uppercase;
    margin-top: 15px;
    font-family: "Oswald";
    font-weight: 500;
    &:hover {
      cursor: pointer;
    }
```

```
}
.qty_change {
  width: 100px;
  height: 30px;
  display: flex;
  justify-content: center;
  border: 1px solid #9f9f9f;
  div {
    padding-left: 22px;
    padding-right: 22px;
    display: flex;
    font-size: 12px;
    justify-content: center;
    align-items: center;
  }
  button {
    &:hover {
      cursor: pointer;
    }
    background-color: transparent;
    border: none;
    display: flex;

    align-items: center;
    color: #9f9f9f;
    font-family: "Oswald";
    font-weight: 400;
    font-size: 20px;
  }
  .qty_inc {
    justify-content: center;
  }
  .qty_dec {
    margin-top: -4px;
  }
}
.center_block {
  display: flex;
  flex-direction: row;
  column-gap: 10px;
}
.cart_header {
  border-bottom: 1px #555555 solid;
  display: flex;
  justify-content: space-between;
  h4 {
    text-transform: uppercase;
    font-size: 30px;
    line-height: 45px;
    letter-spacing: 2px;
    font-weight: 700;
    padding-bottom: 16px;
  }
}
.cart_item {
  display: flex;
  justify-content: space-between;
  border-bottom: 1px #555555 solid;
  padding: 20px 10px;
  img {
    max-width: 60px;
```

```
      height: 80px;
      margin-right: 10px;
    }
  }

  width: 25vw;
  min-width: 400px;
  position: fixed;
  right: 0;
  top: 0;
  background-color: #0a0b0e;
  color: $lightGreeen-color;
  padding: 30px;
  height: 100vh;
  overflow: scroll;
  z-index: 3000;

  .price_item {
    font-size: 16px;
    color: $lightGreeen-color;
  }
}
// CART BLOCK END
</style>

<script>
import axios from "axios";

export default {
  name: "cartItem",
  data() {
    return {
      cart: [],
      flower: [],
      newQtyItem: 0,
    };
  },
  created() {
    window.addEventListener("scroll", this.fixedHeader);
    this.cart = JSON.parse(localStorage.getItem("products in cart")) || [];
    axios.get("/data/flowers.json").then((resp) => {
      this.flowers = resp.data;
    });
  },

  computed: {
    newQty() {
      return this.cart.length;
    },
    total() {
      let sum = 0;
      this.cart.forEach((element) => {
        sum += element.total;
      });
      localStorage.setItem("products in cart", JSON.stringify(this.cart));
      return sum;
    },
  },
  methods: {
    changeProductQty(id, action) {
      const index = this.flowers.findIndex((el) => el.id === id);
```

```
    const indexCart = this.cart.findIndex((el) => el.id === id);
    if (this.flowers[index].id === id) {
     if (action === "inc") {
      this.newQtyItem = this.flowers[index].qty + 1;
     } else {
      if (this.flowers[index].qty >= 2) {
       this.newQtyItem = this.flowers[index].qty - 1;
      } else {
       return false;
      }
     }
     this.flowers[index].qty = this.newQtyItem;
     this.cart[indexCart].qty = this.newQtyItem;
     this.cart[indexCart].total =
      this.cart[indexCart].price * this.newQtyItem;
     // this.calculateTotal();
     localStorage.setItem("products in cart", JSON.stringify(this.cart));
    }
   },
   askProdDel(id) {
    // this.calculateTotal();
    const index = this.cart.findIndex((element) => element.id === id);
    this.cart.splice(index, 1);
    localStorage.setItem("products in cart", JSON.stringify(this.cart));
   },
  },
};
</script>
```

```
<template>
 <header id="header" class="ad_page">
  <!-- MOBILE MENU START -->
  <div id="side-block">
   <div class="menu_wrap">
    <div class="hamb-menu side">
     <button
      class="hamburger hamburger--elastic"
      type="button"
      @click="toggleMenu"
     >
      <span class="hamburger-box">
       <span class="hamburger-inner"></span>
      </span>
     </button>
    </div>
    <div class="logo_wrap">
     <router-link to="/" @click="toggleMenu"
      ><img class="logo" src="../assets/images/logo.svg" alt="logo" />
     </router-link>
    </div>

    <ul class="mobile">
     <li @click="toggleMenu">
      <router-link to="/CatalogView">Каталог</router-link>
     </li>
     <li @click="toggleMenu">
```

```
<router-link to="/DeliveryPaymentView"
          >Доставка і оплата
         </router-link>
        </li>
        <li @click="toggleMenu">
         <router-link to="/AboutView">Про нас</router-link>
        </li>
        <li @click="toggleMenu">
         <router-link to="/ContactsView">Контакти</router-link>
        </li>
        <li @click="toggleMenu">
         <router-link to="/FaqView">FAQ </router-link>
        </li>
       </ul>
       <div class="mail_wrap">
        <router-link
         @click="toggleMenu"
         to="mailto:zakaz@loverflower.by"
         class="bigger link"
         >zakaz@loverflower.by</router-link
        >
        <div class="smaller">Доставка 24/7 по домовленості с оператором</div>
       </div>

       <div class="tel_wrap">
        <router-link
         @click="toggleMenu"
         to="tel:+375 (29) 113-69-69"
         class="bigger"
         >+375 (29) 113-69-69</router-link
        >
        <div class="smaller">Прийом дзвінків цілодобово</div>
       </div>
       <div class="social_wrap">


 <ul class="social_networks">
         <li>
          <router-link
           to="https://www.instagram.com/"
           title="instagram"
           target="_blank"
           rel="noopenner noreferrer"
          >
           <span class="icon-instagram"></span>
          </router-link>
         </li>
         <li>
          <router-link
           to="https://www.whatsapp.com/?lang=uk"
           title="whatsapp"
           target="_blank"
           rel="noopenner noreferrer"
          >
           <span class="icon-whatsapp"></span>
          </router-link>
         </li>

         <li>
          <router-link
```

```
            to="https://www.viber.com/en/"
            title="viber"
            target="_blank"
            rel="noopenner noreferrer"
           >
            <span class="icon-viber"></span>
           </router-link>


     </li>
        </ul>
       </div>
      </div>
     </div>
     <!-- MOBILE MENU END -->
     <div class="page-overlay"></div>

     <div class="container">
      <div class="menu_wrap mobile-version">
       <div class="hamb-menu">
        <button
         class="hamburger hamburger--elastic"
         type="button"
         @click="toggleMenu()"
        >
         <span class="hamburger-box">
          <span class="hamburger-inner"></span>
         </span>
        </button>
       </div>
       <h5>LOVER FLOWER</h5>
       <button @click="toggleCart()" class="cart" type="button">
        <div class="product_qty">{{ this.newQty }}</div>
        <span class="icon-shopping-bag"></span>
       </button>
      </div>
      <div class="menu_wrap main_menu">
       <div class="logo_wrap">
        <router-link to="/"
         ><img class="logo" src="../assets/images/logo.svg" alt="logo" />
        </router-link>
       </div>

       <ul>
        <li><router-link to="/CatalogView">Каталог</router-link></li>
        <li>


<router-link to="/DeliveryPaymentView"
          >Доставка і оплата
         </router-link>
        </li>
        <li>
         <router-link to="/AboutView">Про нас</router-link>
        </li>
        <li><router-link to="/ContactsView">Контакти</router-link></li>
        <li><router-link to="/FaqView">FAQ </router-link></li>
       </ul>

       <div class="right_block_menu">
```

```
        <div class="tel_wrap">
          <a href="tel:+375 (29) 113-69-69">
            <span class="icon-phone"></span> +375 (29) 113-69-69</a
          >
        </div>
        <button @click="toggleCart()" class="cart" type="button">
          <div class="product_qty">{{ this.newQty }}</div>
          <span class="icon-shopping-bag"></span>
        </button>
      </div>
    </div>
  </div>
  <!-- CART BLOCK START -->
  <div class="cart_side">
    <div class="cart_header">
      <h4>Ваша корзина</h4>
      <div @click="toggleCart()" class="close"></div>
    </div>
    <div class="cart_body">



 <cartItem />
      <div class="cart_item"></div>
      <router-link to="/PlacingOrder">
        <button
          class="transparent_btn btn_pay"
          @click="toggleCart()"
          type="button"
        >
          Оформити замовленння
        </button>
      </router-link>
    </div>
  </div>
  <!-- CART BLOCK END -->
 </header>
</template>

<style lang="scss" scoped>
$lightGreeen-color: #43ffd2;
$pastelPinc-color: #d978ac;
.cart {
  .cart_footer {
    .btn_pay {
      width: 100%;
    }
  }
  position: relative;
  height: 55px;
  span {
    font-size: 21px;
  }
}
.transparent_btn {
  margin-top: 20px;
  font-size: 14px;
  font-family: 400;
  padding: 16px 51px;
  // max-width: 320px;
```

```
  width: 100%;
}
.flower_title {
  color: #fff;
  text-transform: uppercase;
  font-size: 14px;
  letter-spacing: 1.2px;
  padding-bottom: 20px;
  max-width: 120px;
  line-height: 16px;
}
.product_qty {
  position: absolute;
  top: 14px;
  left: 36px;
  width: 20px;
  height: 20px;
  font-size: 14px;
  font-weight: 700;
  display: flex;
  justify-content: center;
  align-items: center;
  border-radius: 50%;
  background-color: $lightGreeen-color;
  color: black;
}
.main_menu {
  &.fixed {
    background-color: black;
    box-shadow: 0 0 5px #999;
    left: 0;
    // justify-content: space-around;
    width: 100vw;
    max-width: 100vw;
    min-width: 100vw;
    backdrop-filter: blur(7px);
    box-shadow: 0 0 5px $lightGreeen-color;
    position: fixed;
    opacity: 0.9;
    height: 80px;
    z-index: 7000;
    padding: 6px 120px;
  }
}
.menu_wrap.mobile-version {
  display: none;
}

.hamb-menu {
  display: none;
}
.page-overlay {
  position: fixed;
  width: 100%;
  height: 100%;
  left: 0;
  top: 0;
  background-color: rgba(0, 0, 0, 0.7);
  display: none;
  z-index: 2000;
  &.open {
```

```
    display: block;
  }
}

#header {
  margin: 0;
  padding: 0;
}
#header.ad_page #side-block .menu_wrap ul {
  min-width: 130px;
}
// CART BLOCK START
.cart_side {
  .to_pay {
    text-transform: uppercase;
    margin-top: 25px;
    font-size: 16px;
  }
  .warning {
    color: white;
    text-transform: uppercase;
    font-size: 14px;
    line-height: 17px;
    letter-spacing: 1.2px;
  }
  .delete_btn {
    background-color: transparent;
    border: none;
    color: white;
    border-bottom: 1px solid white;


    text-transform: uppercase;
    margin-top: 15px;
    font-family: "Oswald";
    font-weight: 500;
    &:hover {
      cursor: pointer;
    }
  }
  .qty_change {
    width: 100px;
    height: 30px;
    display: flex;
    justify-content: center;
    border: 1px solid #9f9f9f;
    div {
      padding-left: 22px;
      padding-right: 22px;
      display: flex;
      font-size: 12px;
      justify-content: center;
      align-items: center;
    }
    button {
      &:hover {
        cursor: pointer;
      }
      background-color: transparent;
      border: none;
```

```
    display: flex;

    align-items: center;
    color: #9f9f9f;
    font-family: "Oswald";
    font-weight: 400;
    font-size: 20px;
   }
   .qty_inc {
    justify-content: center;
   }
   .qty_dec {
    margin-top: -4px;
   }
  }
  .center_block {
   display: flex;
   flex-direction: row;
   column-gap: 10px;
  }
  .cart_header {
   border-bottom: 1px #555555 solid;
   display: flex;
   justify-content: space-between;
   h4 {
    text-transform: uppercase;
    font-size: 30px;
    line-height: 45px;
    letter-spacing: 2px;
    font-weight: 700;
    padding-bottom: 16px;
   }
  }
  .cart_item {
   display: flex;
   justify-content: space-between;
   border-bottom: 1px #555555 solid;
   padding: 20px 10px;
   img {
    max-width: 60px;
    height: 80px;
    margin-right: 10px;
   }
  }

  .close {
   width: 30px;
   height: 30px;
   position: relative;
   &:before {
    content: "";
    width: 30px;
    height: 30px;
    position: absolute;
    top: 9px;
    left: 11px;
    border-top: 3px solid $lightGreeen-color;
    transform: rotate(-49deg);
   }
   &:after {
    content: "";
```

```
      width: 30px;
      height: 30px;
      position: absolute;
      top: 8px;
      left: -9px;
      border-top: 3px solid $lightGreeen-color;
      transform: rotate(49deg);
    }
  }
  scrollbar-color: #12130d #f7d794;
  scrollbar-width: thin;
  &::-webkit-scrollbar {
    width: 12px;
  }
  &::-webkit-scrollbar-track {
    background-color: #12130d;
  }
  &::-webkit-scrollbar-thumb {
    background-color: $lightGreeen-color;
    border-radius: 1px;
  }

  width: 350px;
  min-width: 320px;
  position: fixed;
  right: 0;
  top: 0;
  background-color: #0a0b0e;
  color: $lightGreeen-color;
  padding: 30px;
  height: 100vh;
  overflow: auto;
  z-index: 8000;
  -webkit-transform: translateX(350px);
  -ms-transform: translateX(350px);
  transform: translateX(350px);
  -webkit-transition: all 0.7s ease;
  -o-transition: all 0.7s ease;
  transition: all 0.7s ease;
  &.open {
    -webkit-transform: translateX(0);
    -ms-transform: translateX(0);
    transform: translateX(0);
  }
  .price_item {
    font-size: 14px;
  }
}
// CART BLOCK END
#side-block {
  .hamb-menu.side {
    display: block;
    margin-left: 176px;
  }
  .social_wrap {
    background-color: transparent;
    max-width: 200px;
    .social_networks {
      min-width: 139px;
      margin-bottom: -10px;
      background-color: transparent;
```

```
    height: auto;
    display: flex;
    flex-direction: row;
  }
}
.green_text {
  padding-top: 5px;
  margin-left: -20px;
}
.adress {
  max-width: 280px;
  font-size: 15px;
  margin-left: -20px;
  color: #fff;
}
ul li {
  text-decoration: none;
  border-bottom: none;
}
ul {
  min-width: 130px;
}
font-size: 20px;
.mail_wrap,
.tel_wrap {
  align-self: center;
  text-align: center;
  margin-left: -45px;
  color: #fff;
}

.menu_wrap {
  font-size: 20px;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  max-width: 300px;
  ul.mobile {
    min-width: 297px;
    display: flex;
    z-index: 3000;
    flex-direction: column;
    font-size: 18px;
    li {
      margin-bottom: 8px;
      color: $lightGreeen-color;
      border-bottom: none;
    }
    li a {
      font-size: 18px;
      border-bottom: none;
      color: $lightGreeen-color;
    }
  }
}

.hamburger {
  padding: 0;
}
width: 300px;
```

```
  position: fixed;
  left: 0;
  top: 0;
  background-color: #0a0b0e;
  color: $lightGreeen-color;
  border-right: $lightGreeen-color 7px solid;
  box-shadow: 0 0 5px $lightGreeen-color;
  padding: 30px;
  height: 100vh;
  z-index: 3000;
  -webkit-transform: translateX(-320px);
  -ms-transform: translateX(-320px);
  transform: translateX(-320px);
  -webkit-transition: all 0.7s ease;
  -o-transition: all 0.7s ease;
  transition: all 0.7s ease;
  &.open {
    -webkit-transform: translateX(0);
    -ms-transform: translateX(0);
    transform: translateX(0);
  }
}

@media screen and (max-width: 700px) {
  .about_catalog_page .first_line .catalog_desc {
    min-width: 429px;
    max-width: 500px;
    width: 500px;
  }
  .menu_wrap {
    display: none;
  }
  .hamb-menu {
    max-width: 100px;

    // margin-left: 80%;
  }
  .menu_wrap.mobile-version {
    display: block;
    display: flex;

    left: -2px;

    // justify-content: space-around;
    width: 100vw;
    max-width: 100vw;
    min-width: 100vw;
    max-width: 90vw;
    gap: 25px;
    align-items: center;
    justify-content: space-between;
    &.fixed {
      background-color: black;
      box-shadow: 0 0 5px $lightGreeen-color;
      width: 100vw;
      max-width: 100vw;
      backdrop-filter: blur(7px);
      opacity: 0.9;
    }

    h5 {
```

```
      font-family: "Cormorant";
      font-weight: 300;
      font-size: 20px;
      max-width: 100px;
      line-height: 24px;
      padding-top: 5px;
      letter-spacing: 1.2px;
    }
    .hamb-menu {
      display: block;
    }
  }
}
</style>
<script>
// import {  } from "@babel/types";
import axios from "axios";
import cartItem from "@/components/cartItem.vue";

export default {
  name: "MainHeader",
  components: {
    cartItem,
  },
  data() {
    return {
      cart: [],
      flower: [],
      newQtyItem: 0,
    };
  },
  created() {
    window.addEventListener("scroll", this.fixedHeader);
    this.cart = JSON.parse(localStorage.getItem("products in cart")) || [];
    axios.get("/data/flowers.json").then((resp) => {
      this.flowers = resp.data;
    });
  },

  computed: {
    newQty() {
      return this.cart.length;
    },
    total() {
      let sum = 0;
      this.cart.forEach((element) => {
        sum += element.total;
      });
      return sum;
    },
  },
  methods: {
    changeProductQty(id, action) {
      const index = this.flowers.findIndex((el) => el.id === id);
      const indexCart = this.cart.findIndex((el) => el.id === id);
      if (this.flowers[index].id === id) {
        if (action === "inc") {
          this.newQtyItem = this.flowers[index].qty + 1;
        } else {
          if (this.flowers[index].qty >= 2) {
            this.newQtyItem = this.flowers[index].qty - 1;
```

```
        } else {
          return false;
        }
      }
      this.flowers[index].qty = this.newQtyItem;
      this.cart[indexCart].qty = this.newQtyItem;
      this.cart[indexCart].total =
        this.cart[indexCart].price * this.newQtyItem;
      // this.calculateTotal();
      localStorage.setItem("products in cart", JSON.stringify(this.cart));
    }
  },
  askProdDel(id) {
    // this.calculateTotal();
    const index = this.cart.findIndex((element) => element.id === id);
    this.cart.splice(index, 1);
    localStorage.setItem("products in cart", JSON.stringify(this.cart));
  },
  addToCart() {
    this.cart = JSON.parse(localStorage.getItem("products in cart")) || [];
  },
  fixedHeader() {
    if (window.scrollY > 90) {
      document
        .querySelector(" .menu_wrap.mobile-version")
        .classList.add("fixed");
      document.querySelector(".main_menu").classList.add("fixed");
    } else {
      document
        .querySelector(" .menu_wrap.mobile-version")
        .classList.remove("fixed");
      document.querySelector(".main_menu").classList.remove("fixed");
    }
  },

  toggleCart() {
    // this.calculateTotal();
    document.querySelector(".cart_side").classList.toggle("open");
  },
  toggleMenu() {
    document.querySelector(".hamburger").classList.toggle("is-active");
    document.querySelector("#side-block").classList.toggle("open");
    document.querySelector(".page-overlay").classList.toggle("open");
    document.querySelector("body").classList.toggle("lock");
  },
 },
};
</script>
<!-- <template>
 <div>
  <div v-for="flower in flowers" :key="flower">
   <div class="flag_wrap">
    <div v-if="flower.isSale" class="sale">SALE</div>
    <div v-if="flower.isNew" class="new">NEW</div>
   </div>

   <div class="flower_card">
    <img
     :src="require(`@/assets/images/flowers/${flower.images[0]}`)"
     class="card-img-top"
     :alt="flower.title"
```

```
      />
      <div class="bouquet_name">{{ flower.title }}</div>


 <div class="price">
       {{ flower.price }} UAN
       <span class="old_price" v-if="flower.isSale"
        >{{ flower.old_price }} UAN</span
       >
      </div>
      <button
       class="transparent_btn"
       type="button"
       @click="addToCart(flower.id, flower.price, flower.title)"
      >
       У корзину
      </button>
     </div>
    </div>
  </div>
</template>

<script>
import axios from "axios";
export default {
 name: "oneProduct",
 data() {
  return {
   cart: [],
   flowers: [],
   newQtyItem: 0,
  };
 },
 created() {
  axios.get("/data/flowers.json").then((resp) => {
   this.flowers = resp.data;
  });
  this.cart = JSON.parse(localStorage.getItem("products in cart")) || [];
 },
 methods: {
  addToCart(id, price, title) {
   let qty = 1;
   if (this.cart.find((el) => el.id === id) === undefined) {
    this.cart.push({
     name: title,
     id: id,
     qty: qty,
     isBuy: false,
     price: price,
     total: parseFloat((qty * price).toFixed(2)),
    });
    localStorage.setItem("products in cart", JSON.stringify(this.cart));
    this.cart = JSON.parse(localStorage.getItem("products in cart")) || [];
   } else {
    if (this.cart.findIndex((el) => el.id === id) !== undefined) {
     const prodIndex = this.cart.findIndex((el) => el.id === id);
     const newQty = this.cart[prodIndex].qty + 1;
     this.cart[prodIndex].qty = newQty;
     this.cart[prodIndex].total = parseFloat(
      (newQty * this.cart[prodIndex].price).toFixed(2)
```

```
        );
      }

      localStorage.setItem("products in cart", JSON.stringify(this.cart));
      this.cart = JSON.parse(localStorage.getItem("products in cart")) || [];
    }
   },
  },
};
</script> -->
<template>
  <div class="track-container">
    <div class="track" ref="_vpcTrack"></div>
    <div class="track-highlight" ref="trackHighlight"></div>
    <button class="track-btn track1" ref="track1"></button>
    <button class="track-btn track2" ref="track2"></button>
    <div class="price_range">ЦІНА: {{ minValue }} - {{ maxValue }} UAN</div>
  </div>
</template>

<script>
export default {
  name: "PriceRangeSlider",

  props: {
    trackHeight: {
      type: Number,
      deafult: 1,
    },
  },

  data() {
    return {
      min: 50,
      max: 10000,
      minValue: 50,
      maxValue: 10000,
      step: 5,
      totalSteps: 0,
      percentPerStep: 1,
      trackWidth: null,
      isDragging: false,
      pos: {
        curTrack: null,
      },
    };
  },

  methods: {
    moveTrack(track, ev) {
      let percentInPx = this.getPercentInPx();

      let trackX = Math.round(
        this.$refs._vpcTrack.getBoundingClientRect().left
      );
      let clientX = ev.clientX;
      let moveDiff = clientX - trackX;

      let moveInPct = moveDiff / percentInPx;
      // console.log(moveInPct)
```

```
  if (moveInPct < 1 || moveInPct > 100) return;
  let value =
    Math.round(moveInPct / this.percentPerStep) * this.step + this.min;
  if (track === "track1") {
    if (value >= this.maxValue - this.step) return;
    this.minValue = value;
  }

  if (track === "track2") {
    if (value <= this.minValue + this.step) return;
    this.maxValue = value;
  }

  this.$refs[track].style.left = moveInPct + "%";
  this.setTrackHightlight();
  this.$emit("change-range", this.minValue, this.maxValue);
},
mousedown(ev, track) {
  if (this.isDragging) return;
  this.isDragging = true;
  this.pos.curTrack = track;
},

touchstart(ev, track) {
  this.mousedown(ev, track);
},

mouseup() {
  if (!this.isDragging) return;
  this.isDragging = false;
},

touchend(ev, track) {
  this.mouseup(ev, track);
},

mousemove(ev, track) {
  if (!this.isDragging) return;
  this.moveTrack(track, ev);
},

touchmove(ev, track) {
  this.mousemove(ev.changedTouches[0], track);
},

valueToPercent(value) {
  return ((value - this.min) / this.step) * this.percentPerStep;
},

setTrackHightlight() {
  this.$refs.trackHighlight.style.left =
    this.valueToPercent(this.minValue) + "%";
  this.$refs.trackHighlight.style.width =
    this.valueToPercent(this.maxValue) -
    this.valueToPercent(this.minValue) +
    "%";
},

getPercentInPx() {
  let trackWidth = this.$refs._vpcTrack.offsetWidth;
  let oneStepInPx = trackWidth / this.totalSteps;
```

```
    // 1 percent in px
    let percentInPx = oneStepInPx / this.percentPerStep;

    return percentInPx;
  },

  setClickMove(ev) {
    let track1Left = this.$refs.track1.getBoundingClientRect().left;
    let track2Left = this.$refs.track2.getBoundingClientRect().left;
    // console.log('track1Left', track1Left)
    if (ev.clientX < track1Left) {
      this.moveTrack("track1", ev);
    } else if (ev.clientX - track1Left < track2Left - ev.clientX) {
      this.moveTrack("track1", ev);
    } else {
      this.moveTrack("track2", ev);
    }
  },
},

mounted() {
  // calc per step value
  this.totalSteps = (this.max - this.min) / this.step;

  // percent the track button to be moved on each step
  this.percentPerStep = 100 / this.totalSteps;
  // console.log('percentPerStep', this.percentPerStep)

  // set track1 initilal
  document.querySelector(".track1").style.left =
    this.valueToPercent(this.minValue) + "%";
  // track2 initial position
  document.querySelector(".track2").style.left =
    this.valueToPercent(this.maxValue) + "%";
  // set initila track highlight
  this.setTrackHightlight();

  var self = this;

  ["mouseup", "mousemove"].forEach((type) => {
    document.body.addEventListener(type, (ev) => {
      // ev.preventDefault();
      if (self.isDragging && self.pos.curTrack) {
        self[type](ev, self.pos.curTrack);
      }
    });
  });

  [
    "mousedown",
    "mouseup",
    "mousemove",
    "touchstart",
    "touchmove",
    "touchend",
  ].forEach((type) => {
    document.querySelector(".track1").addEventListener(type, (ev) => {
      ev.stopPropagation();
      self[type](ev, "track1");
    });
```

```
    document.querySelector(".track2").addEventListener(type, (ev) => {
      ev.stopPropagation();
      self[type](ev, "track2");
    });
  });

  // on track clik
  // determine direction based on click proximity
  // determine percent to move based on track.clientX - click.clientX
  document.querySelector(".track").addEventListener("click", function (ev) {
    ev.stopPropagation();
    self.setClickMove(ev);
  });

  document
    .querySelector(".track-highlight")
    .addEventListener("click", function (ev) {
      ev.stopPropagation();
      self.setClickMove(ev);
    });
 },
};
</script>
<template>
 <!-- CART BLOCK START -->
 <div class="cart_side">
  <div class="cart_header">
   <h4>Ваша корзина</h4>
   <div @click="toggleCart()" class="close"></div>
  </div>
  <div class="cart_body">
   <div class="cart_item" v-for="flower in cart" :key="flower">
    <div class="center_block">
     <img
      :src="require(`@/assets/images/flowers/${flower.id}.1.jpg`)"
      class="card-img-top"
      :alt="flower.title"
     />

     <div class="wrap_qty_title">
      <div class="flower_title">
       {{ flower.name }}
      </div>
      <div class="qty_change">
       <button
        type="button"
        class="_inc"
        @click="changeProductQty(flower.id, 'inc')"
       >
        +
       </button>
       <div>{{ flower.qty }}</div>
       <button
        type="button"
        class="qty_dec"
        @click="changeProductQty(flower.id, 'dec')"
       >
        -
       </button>
      </div>
     </div>
```

```
        </div>
        <div class="total_block">
          <div class="price_item">{{ flower.total }} UAN</div>
          <button
            type="button"
            class="delete_btn"
            @click="askProdDel(flower.id)"
          >
            видалити


        </button>
        </div>
      </div>
    </div>
    <div class="cart_footer">
      <div class="to_pay">До сплати: {{ this.total }} UAN</div>
      <div class="warning">
        Щоб дізнатися про вартість доставки, перейдіть до оформлення замовлення.
      </div>

      <router-link to="/PlacingOrder">
        <button
          class="transparent_btn btn_pay"
          @click="toggleCart()"
          type="button"
        >
          Оформити замовленння
        </button>
      </router-link>
    </div>
  </div>
  <!-- CART BLOCK END -->
</template>
<script>
import axios from "axios";
export default {
  name: "shopCart",

  props: {},
  data() {
    return {
      cart: [],
      flowers: [],
    };
  },
  created() {
    this.cart = JSON.parse(localStorage.getItem("products in cart")) || [];
    axios.get("/data/flowers.json").then((resp) => {
      this.flowers = resp.data;
    });
  },
  computed: {
    newQty() {
      return this.cart.length;
    },
    total() {
      let sum = 0;
      this.cart.forEach((element) => {
        sum += element.total;
```

```
    });
    return sum;
   },
  },
};
</script>

<style lang="scss" scoped></style>
<template>
 <div class="carousel">
  <carousel
   :items-to-show="3"
   :settings="sliderSettings"
   :breakpoints="breakpoints"
  >
   <slide v-for="flower in flowers" :key="flower">
    <!-- <oneProduct /> -->
    <div class="flag_wrap">
     <div v-if="flower.isSale" class="sale">SALE</div>
     <div v-if="flower.isNew" class="new">NEW</div>
    </div>

    <div class="flower_card">
     <img
      :src="require(`@/assets/images/flowers/${flower.images[0]}`)"
      class="card-img-top"
      :alt="flower.title"
     />
     <div class="bouquet_name">{{ flower.title }}</div>
     <div class="price">
      {{ flower.price }} UAN
      <span class="old_price" v-if="flower.isSale"
       >{{ flower.old_price }} UAN</span
      >
     </div>
     <button
      class="transparent_btn"
      type="button"
      @click="addToCart(flower.id, flower.price, flower.title)"
     >
      У корзину
     </button>
    </div>
   </slide>

   <template #addons>
    <navigation />
    <pagination />
   </template>
  </carousel>
 </div>
</template>

<script>
// If you are using PurgeCSS, make sure to whitelist the carousel CSS classes
import "vue3-carousel/dist/carousel.css";
import axios from "axios";
import { Carousel, Slide, Pagination, Navigation } from "vue3-carousel";
// import oneProduct from "@/components/oneProduct.vue";
export default {
 name: "sliderFavorite",
```

```
components: {
  // oneProduct,
  Carousel,
  Slide,
  Pagination,
  Navigation,
},
data() {
  return {
    cart: [],
    flowers: [],
    newQtyItem: 0,
    sliderSettings: {
      itemsToShow: 3,
      // transition: 800,
      // itemsToScroll: 1,
      // infiniteLoop: true,
      // snapAlign: "start",
      // wrapAround: true,
    },

    breakpoints: {
      1024: {
        itemsToShow: 3,
      },
      700: {
        itemsToShow: 2,
      },
      480: {
        itemsToShow: 1,
      },

      // 1024 and up
    },
  };
},
created() {
  axios.get("/data/flowers.json").then((resp) => {
    this.flowers = resp.data;
  });
  this.cart = JSON.parse(localStorage.getItem("products in cart")) || [];
},
methods: {
  addToCart(id, price, title) {
    let qty = 1;
    if (this.cart.find((el) => el.id === id) === undefined) {
      this.cart.push({
        name: title,
        id: id,
        qty: qty,
        isBuy: false,
        price: price,
        total: parseFloat((qty * price).toFixed(2)),
      });
      localStorage.setItem("products in cart", JSON.stringify(this.cart));
      this.cart = JSON.parse(localStorage.getItem("products in cart")) || [];
    } else {
      if (this.cart.findIndex((el) => el.id === id) !== undefined) {
        const prodIndex = this.cart.findIndex((el) => el.id === id);
        const newQty = this.cart[prodIndex].qty + 1;
        this.cart[prodIndex].qty = newQty;
```

```
        this.cart[prodIndex].total = parseFloat(
          (newQty * this.cart[prodIndex].price).toFixed(2)
        );
      }

      localStorage.setItem("products in cart", JSON.stringify(this.cart));
      this.cart = JSON.parse(localStorage.getItem("products in cart")) || [];
    }
   },
  },
};
</script>

<style lang="scss" scoped>
$lightGreeen-color: #43ffd2;
$pastelPinc-color: #d978ac;
$pastelPinc-colorSpot: #d978ac8b;
$lightGreeen-colorSpot: #43ffd399;

@mixin triangle {
  content: "";
  position: absolute;
  top: 20px;
  left: -32px;
  transform: rotate(323deg);
  width: 0;
  height: 0;
  border-left: 5px solid transparent;
  border-right: 5px solid transparent;
  border-top: 5px solid $lightGreeen-color;
}

.flower_card {
  position: relative;
}

.flag_wrap {
  display: flex;
  flex-direction: column;
  row-gap: 5px;
  position: absolute;
  top: 10px;
  left: 315px;
}

.sale,
.new {
  position: absolute;
  top: 10px;
  left: 315px;
  padding-top: 13px;
  text-align: center;
  width: 60px;
  z-index: 3;
  height: 60px;
  border-radius: 50%;
  color: black;
  font-weight: 300;
  font-size: 20px;
}
.sale {
```

```
  background-color: #31985a;
}
.new {
  background-color: #d978ac;
}
.flag_wrap {
  .sale,
  .new {
    position: static;
  }
}
img {
  min-width: 300px;
  max-width: 350px;
  max-height: 450px;
  padding-bottom: 10px;
  filter: grayscale(100%);
  &:hover,
  &:focus {
    filter: grayscale(0%);
  }
}

.price {
  padding-bottom: 20px;
  font-weight: 700;
  letter-spacing: 0.9px;
}
.bouquet_name {
  text-transform: uppercase;
  font-size: 20px;
  flex-wrap: 400;
  letter-spacing: 1.7px;
  padding-bottom: 10px;
}
.price {
  font-size: 14px;
  color: #fff;
  font-family: "Oswald";
  font-weight: 700;
  margin-bottom: 0;
  line-height: 16px;
  .old_price {
    color: #939393;
    font-weight: 300;
    text-decoration: line-through;
    padding-left: 10px;
    line-height: 16px;
  }
}
.carousel__pagination {
  display: none;
}
.carousel__slide {
  width: 200px !important;
}

.carousel__slide[data-v-6b2292e8] {
  width: 350px !important;
  margin-right: 30px;
  margin-left: 10px;
```

```
}
.flower_card {
  text-align: left;
}

// RESPONSIVE SLIDER
@media screen and (max-width: 860px) {
  .flower_card {
    img {
      min-width: 280px;
      max-width: 283px !important;
      width: 280px !important;
      height: 394px;
    }
  }
  .carousel__slide {
    margin-right: 1px !important;
    margin-left: -10px !important ;

    .new,
    .sale {
      left: 288px;
    }
    .flag_wrap {
      left: 288px;
    }
  }}</style>
```