Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

Допущено до захисту
Завідувач кафедри комп’ютерних наук
доктор технічних наук, професор
Андрій БОНДАРЧУК
« 01 » червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»
Спеціальність 122 Комп’ютерні науки
Освітня програма 122.00.01 Інформатика

Тема: ІНТЕРАКТИВНИЙ ВЕБСАЙТ КОМЕРЦІЙНОЇ РЕАЛІЗАЦІЇ КВІТІВ

Виконав
студент групи ІНб-2-22-4.0д
Лаврінець Максим Олексійович
_____
(підпис)

Науковий керівник
Кандидат технічних наук,
доцент
Яскевич В.О.
_____
(підпис)

Київ – 2026

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

«Затверджую»
Завідувач кафедри комп'ютерних наук
Доктор технічних наук, професор
Андрій БОНДАРЧУК
«31» 10. 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІНУ РОБОТУ БАКАЛАВРА
«ІНТЕРАКТИВНИЙ ВЕБСАЙТ КОМЕРЦІЙНОЇ РЕАЛІЗАЦІЇ КВІТІВ»

Виконавець – студент групи ІНб-2-22-4.0д,
спеціальності 122 Комп'ютерні наук,
освітньої програми 122.00.01 Інформатика
Лаврінець Максим Олексійович

1. Вихідні дані: Науково-технічна література, сучасні веб-технології та програмні засоби розробки інтерактивних e-commerce систем. Матеріали щодо проектування клієнт-серверних веб-додатків, адаптивного дизайну, frontend та backend розробки, систем керування базами даних та сучасних технологій електронної комерції.
2. Основні завдання: Провести аналіз сучасних e-commerce вебсайтів та обґрунтувати вибір технологій для розробки веб-додатку. Спроектувати структуру інтерактивного вебсайту та архітектуру взаємодії між frontend і backend частинами системи. Реалізувати адаптивний користувацький інтерфейс, каталог товарів, систему навігації, кошика та оформлення замовлення. Розробити структуру бази даних SQLite, REST API та серверну частину із використанням Flask. Провести тестування функціоналу веб-додатку та підготувати пояснювальну записку, графічні матеріали і презентацію до захисту кваліфікаційної роботи.
3. Пояснювальна записка: Обсяг приблизно 55 стор формату А4 комп'ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри
4. Графічні матеріали: Комп'ютерна презентація доповіді.
5. Додатки: фрагменти програмного коду вебсайту комерційної реалізації квітів (React-компоненти інтерфейсу, модулі каталогу товарів, кошика та оформлення замовлення, реалізація серверної частини Flask і структура бази даних SQLite).
6. Строк подання роботи на кафедру: *«01» червня 2026р.*

Науковий керівник
к.т.н., доцент
_____ Яскевич В.О.
31. 10. 2025р дата

Виконавець:
Лаврінець М.О.
31. 10. 2025р дата

КАЛЕНДАРНИЙ ПЛАН

№	Етапи виконання роботи	Термін виконання	Примітка
1	Затвердження теми кваліфікаційної роботи бакалавра	31.10.25	Виконано
2	Аналіз предметної області, дослідження сучасних вебсайтів електронної комерції, формування цілей і завдань	01.11.25 – 11.01.26	Виконано
3	Аналіз технологій frontend та backend розробки, обґрунтування вибору технологічного стеку проєкту	26.01.26 – 15.03.26	Виконано
4	Дослідження та проєктування структури вебсайту, навігації, архітектури системи та структури бази даних	16.03.26 – 31.03.26	Виконано
5	Розробка клієнтської частини вебдодатку засобами React та Tailwind CSS	01.04.26 – 15.04.26	Виконано
6	Тестування функціоналу вебдодатку: каталог товарів, кошик, оформлення замовлення	16.04.26 – 30.04.26	Виконано
7	Впровадження серверної частини, REST API та інтеграція з базою даних SQLite	01.05.26 – 15.05.26	Виконано
8	Підготовка пояснювальної записки, графічних матеріалів, додатків	25.05.25 – 12.06.26	Виконано
9	Захист кваліфікаційної роботи	19.06.26	

«Затверджую»

Науковий керівник

к.т.н., доцент, Яскевич В.О.

Індивідуальний план склав

Лаврінець М.О.

РЕФЕРАТ

Кваліфікаційна робота бакалавра: 55 с., 12 рис., 7 табл., 45 джерел та 1 додаток.

Актуальність. Сфера електронної комерції продовжує активно розвиватися та охоплює все більше напрямів торгівлі. Одним із таких напрямів є реалізація квіткової продукції через мережу Інтернет. Для покупця важливими є зручний пошук товарів, наочне представлення продукції, швидке оформлення замовлення та можливість перегляду інформації про товари на різних типах пристроїв.

Багато невеликих квіткових магазинів використовують готові конструктори сайтів або системи керування контентом, які мають обмеження щодо зміни функціоналу та адаптації інтерфейсу до власних потреб. У зв’язку з цим актуальною є розробка інтерактивного вебсайту комерційної реалізації квітів із сучасною клієнт-серверною архітектурою та адаптивним користувацьким інтерфейсом.

Об’єкт дослідження: інформаційна система електронної комерції для реалізації квіткової продукції через мережу Інтернет.

Предмет дослідження: методи, моделі та технології побудови інформаційних систем електронної комерції, зокрема організація взаємодії клієнтської та серверної частин, керування даними та реалізація функціональних модулів вебсайту.

Мета роботи: розробка інтерактивного вебсайту для комерційної реалізації квітів із сучасним адаптивним інтерфейсом, динамічним каталогом товарів, системою категорій та модулем оформлення замовлень.

Завдання роботи:

1. Проаналізувати предметну область електронної комерції та сучасні вебсайти з продажу квіткової продукції.
2. Визначити функціональні та нефункціональні вимоги до інформаційної системи комерційної реалізації квітів.

3. Спроєктувати архітектуру вебсайту, структуру бази даних та взаємодію клієнтської і серверної частин системи.

4. Розробити клієнтську частину вебсайту з використанням React та забезпечити реалізацію каталогу товарів, пошуку, кошика та оформлення замовлень.

5. Реалізувати серверну частину вебсайту на основі Flask та забезпечити взаємодію з базою даних SQLite через REST API.

6. Провести тестування розробленої системи та оцінити результати її функціонування.

Основні результати. Розроблено інтерактивний вебсайт комерційної реалізації квітів із використанням технологій React, Tailwind CSS, Flask та SQLite. Реалізовано каталог товарів із системою категорій, пошуком і сортуванням, сторінки окремих товарів, кошик покупця, модуль оформлення замовлень, форму індивідуального замовлення букета, особистий кабінет користувача та адміністративну панель.

Для обміну даними між клієнтською та серверною частинами використано REST API. Зберігання інформації про користувачів, товари, замовлення та індивідуальні заявки забезпечується базою даних SQLite.

Практичне значення дослідження. Розроблений вебсайт може бути використаний як основа для створення повноцінного інтернет-магазину квіткової продукції. Запропоновані архітектурні та програмні рішення можуть застосовуватися під час розробки інших вебсистем електронної комерції.

Ключові слова: електронна комерція, вебсайт, React, Flask, SQLite, REST API, frontend, backend, адаптивний дизайн, інтернет-магазин квітів

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface, програмний інтерфейс застосунку.

CRUD – Create, Read, Update, Delete, базові операції роботи з даними.

CSS – Cascading Style Sheets, каскадні таблиці стилів.

DB – Database, база даних.

DBMS (СУБД) – система управління базами даних.

Flask – мікрофреймворк для розробки вебдодатків мовою Python.

HTML – HyperText Markup Language, мова розмітки вебсторінок.

HTTP – HyperText Transfer Protocol, протокол передачі гіпертексту.

JSON – JavaScript Object Notation, формат обміну даними.

JS (JavaScript) – мова програмування для створення інтерактивних вебінтерфейсів.

JSX – JavaScript XML, синтаксичне розширення JavaScript для React.

ORM – Object Relational Mapping, технологія взаємодії об’єктів програми з базою даних.

React – JavaScript-бібліотека для розробки користувацьких інтерфейсів.

REST – Representational State Transfer, архітектурний стиль побудови вебсервісів.

REST API – програмний інтерфейс взаємодії між клієнтською та серверною частинами системи на основі REST.

SPA – Single Page Application, односторінковий вебдодаток.

SQL – Structured Query Language, мова структурованих запитів до баз даних.

SQLite – вбудована реляційна система управління базами даних.

SQLAlchemy – ORM-бібліотека для роботи з базами даних у Python.

UI – User Interface, користувацький інтерфейс.

UML – Unified Modeling Language, мова моделювання програмних систем.

URL – Uniform Resource Locator, уніфікований покажчик ресурсу.

UX – User Experience, користувацький досвід взаємодії користувача із системою.

Vite – інструмент для розробки та збірки сучасних вебдодатків.

WWW – World Wide Web, глобальна система вебресурсів мережі Інтернет.

ЗМІСТ

ВСТУП

Актуальність теми. Електронна комерція залишається одним із найбільш поширених напрямів використання інформаційних технологій. Значна кількість товарів і послуг реалізується через мережу Інтернет, що зумовлює потребу у створенні сучасних вебсайтів для зручної взаємодії між продавцем і покупцем [25, 26].

Одним із напрямів електронної комерції є реалізація квіткової продукції. Під час вибору квітів важливу роль відіграє візуальне представлення товару, швидкий доступ до інформації про букет, зручна навігація каталогом та можливість оперативного оформлення замовлення.

Багато невеликих квіткових магазинів використовують готові конструктори сайтів або системи керування контентом. Проте такі рішення часто мають обмеження щодо зміни функціоналу та інтеграції додаткових сервісів. Використання сучасних вебтехнологій дозволяє створити вебдодаток із клієнт-серверною архітектурою, адаптивним інтерфейсом та зручною взаємодією користувача із системою [16, 21].

У зв’язку з цим актуальною є розробка інтерактивного вебсайту комерційної реалізації квітів, який забезпечує зручний перегляд каталогу товарів, оформлення замовлень та керування інформацією про продукцію.Мета роботи: розробка інтерактивного вебсайту для комерційної реалізації квітів із сучасним адаптивним інтерфейсом, динамічним каталогом товарів, системою категорій та модулем оформлення замовлень.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

- проаналізувати предметну область електронної комерції та сучасні вебсайти з продажу квіткової продукції;
- визначити функціональні та нефункціональні вимоги до інформаційної системи комерційної реалізації квітів;
- спроєктувати архітектуру вебсайту, структуру бази даних та взаємодію клієнтської і серверної частин системи;

- розробити клієнтську частину вебсайту з використанням React та реалізувати каталог товарів, пошук, кошик і оформлення замовлень;
- реалізувати серверну частину вебсайту на основі Flask та забезпечити взаємодію з базою даних SQLite через REST API;
- провести тестування розробленої системи та оцінити результати її функціонування.

Об’єкт дослідження: інформаційна система електронної комерції для реалізації квіткової продукції через мережу Інтернет.

Предмет дослідження: методи, моделі та технології побудови інформаційних систем електронної комерції, зокрема організація взаємодії клієнтської та серверної частин, керування даними та реалізація функціональних модулів вебсайту.

Методи дослідження. Під час виконання роботи використано методи системного аналізу для дослідження предметної області та формування вимог до інформаційної системи, методи проєктування баз даних для побудови структури зберігання інформації, принципи компонентного програмування для реалізації клієнтської частини вебсайту та методи функціонального тестування для перевірки коректності роботи системи [6, 33, 35]. Для розробки програмного забезпечення використано React, Flask, SQLite та Tailwind CSS [16, 19, 21, 24].

Практичне значення роботи полягає у створенні інтерактивного вебсайту для реалізації квіткової продукції через мережу Інтернет. Розроблене програмне рішення може бути використане як основа для створення повноцінного інтернет-магазину квітів, а також для подальшого розширення функціональних можливостей системи.

Структура роботи. Робота складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі проведено аналіз предметної області та розглянуто сучасні технології веброзробки. У другому розділі описано процес проєктування структури вебсайту, архітектури системи та бази даних. У третьому розділі наведено результати реалізації вебсайту та тестування його основних функцій.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СУЧАСНИХ ВЕБТЕХНОЛОГІЙ

Розробка вебсайту комерційної реалізації квітів потребує попереднього аналізу предметної області, дослідження існуючих рішень та визначення вимог до майбутньої системи. Перед початком програмної реалізації необхідно визначити особливості організації продажу квіткової продукції через мережу Інтернет, проаналізувати функціональні можливості сучасних вебресурсів та обрати технології, які забезпечать реалізацію необхідного функціоналу.

У цьому розділі виконано аналіз предметної області, розглянуто вебсайти, що працюють у сфері продажу квіткової продукції, визначено функціональні та нефункціональні вимоги до системи, проведено порівняння сучасних технологій веброзробки та обґрунтовано вибір технологічного стеку для реалізації вебсайту BloomShop.

1.1 Аналіз предметної області та чинних галузевих рішень

Реалізація квіткової продукції через мережу Інтернет має власні особливості порівняно з іншими напрямами електронної комерції. Під час вибору букета або квіткової композиції покупець насамперед оцінює зовнішній вигляд товару, тому важливе значення мають якісні фотографії, зрозуміла структура каталогу та швидкий доступ до інформації про продукцію [25, 26]. Від зручності навігації та процесу оформлення замовлення значною мірою залежить загальне враження користувача від вебресурсу.

Для квіткового магазину вебсайт виконує не лише роль каталогу товарів, а й виступає основним інструментом взаємодії з клієнтами. Користувач повинен мати можливість швидко знайти потрібний товар, переглянути його характеристики, додати до кошика та оформити замовлення без виконання великої кількості додаткових дій. Саме тому під час розробки подібних систем особлива увага приділяється організації каталогу продукції, реалізації пошуку, адаптивності інтерфейсу та зручності користування вебсайтом.

На сьогодні існує велика кількість вебресурсів, які спеціалізуються на продажу букетів, квіткових композицій, кімнатних рослин та супутніх товарів. Незважаючи на схожий набір базових функцій, кожен вебсайт має власні підходи до організації каталогу, представлення товарів та процесу оформлення замовлень. Для визначення вимог до розроблюваного вебсайту було проведено аналіз кількох популярних рішень, що працюють у даній сфері.

Одним із таких ресурсів є Florium [42]. Вебсайт спеціалізується на продажу квітів, саджанців та декоративних рослин. Основною особливістю ресурсу є широкий асортимент продукції та детальна структура каталогу. Користувачам доступний пошук товарів, перегляд характеристик і оформлення замовлень через вебінтерфейс.

Перевагою Florium є значна кількість товарних позицій та докладний опис продукції. Водночас велика кількість інформаційних блоків на окремих сторінках може ускладнювати навігацію та пошук потрібного товару, особливо під час використання мобільних пристроїв.

Наступним прикладом є вебсайт Dicentra [43], орієнтований на продаж букетів і флористичних композицій. У даному рішенні значна увага приділяється візуальному представленню продукції. Для демонстрації товарів використовуються великі фотографії, рекламні блоки та акцентні елементи інтерфейсу.

Перевагою Dicentra є сучасний дизайн і якісне представлення товарів. Разом із цим велика кількість графічних елементів може впливати на швидкість завантаження сторінок. Частина функціональних можливостей також потребує додаткових переходів між сторінками, що збільшує кількість дій користувача під час оформлення замовлення.

Для аналізу було розглянуто і вебсайт DonPion [44], який спеціалізується на продажу букетів та подарункових наборів. Ресурс пропонує базовий набір функцій електронної комерції, зокрема перегляд каталогу, вибір товарів за категоріями та оформлення замовлення.

Позитивною особливістю DonPion є проста структура каталогу та зрозумілий процес придбання товарів. Разом із тим можливості пошуку та фільтрації продукції реалізовані обмежено, що може створювати незручності під час роботи з великим асортиментом товарів.

Проведений аналіз показав, що більшість вебсайтів квіткової продукції використовують подібний набір функціональних можливостей. До них належать каталог товарів, сторінки окремих товарів, кошик покупця та оформлення замовлень. Відмінності між ресурсами переважно стосуються організації інтерфейсу, структури каталогу та зручності взаємодії користувача із системою.

Для розроблюваного вебсайту BloomShop доцільно врахувати позитивні сторони розглянутих рішень та уникнути виявлених недоліків. Зокрема, система повинна забезпечувати структурований каталог продукції, підтримувати пошук і фільтрацію товарів, надавати можливість швидкого оформлення замовлення та коректно працювати на різних типах пристроїв. Крім того, важливим елементом є наявність адміністративної панелі для керування товарами та замовленнями.

Таблиця 1.1

Порівняльна характеристика вебсайтів реалізації квіткової продукції

Критерій порівняння	Florium	Dicentra	DonPion
Основний напрям діяльності	Продаж квітів, рослин та саджанців	Продаж букетів і квіткових композицій	Продаж букетів та подарункових наборів
Каталог продукції	Розгалужений каталог із великою кількістю категорій	Каталог із акцентом на візуальне представлення товарів	Каталог із базовим набором категорій
Пошук та навігація	Реалізовано пошук і категорії	Реалізовано пошук і категорії	Реалізовано категорії, пошук обмежений
Фільтрація товарів	Наявна	Частково реалізована	Практично відсутня
Представлення товарів	Детальний опис та характеристики	Великі фотографії та короткий опис	Спрощений опис продукції
Процес оформлення замовлення	Стандартний	Стандартний	Спрощений
Адаптація до мобільних пристроїв	Підтримується	Підтримується	Підтримується
Виявлені недоліки	Перевантаженість інформаційними блоками	Велика кількість графічних елементів	Обмежені можливості пошуку та фільтрації

Порівняння розглянутих вебресурсів показало, що всі вони забезпечують базові можливості електронної комерції, необхідні для продажу квіткової продукції через мережу Інтернет. Разом із тим кожне рішення має власні особливості та певні обмеження. Результати аналізу були враховані під час проєктування вебсайту BloomShop. На їх основі визначено функціональні можливості майбутньої системи, зокрема реалізацію структурованого каталогу товарів, пошуку та фільтрації продукції, кошика покупця, оформлення замовлень, особистого кабінету користувача та адміністративної панелі. Сформовані висновки стали основою для визначення функціональних і нефункціональних вимог до системи, які розглядаються в наступному підрозділі.

1.2 Обгрунтування вимог до інформаційної системи комерційної реалізації квітів

Після аналізу предметної області та існуючих вебресурсів було визначено перелік можливостей, які повинні бути реалізовані у вебсайті BloomShop. Визначення вимог є необхідним етапом проєктування, оскільки саме на їх основі формується структура системи, обираються технології розробки та реалізується функціонал вебдодатку.

Під час формування вимог враховувалися особливості продажу квіткової продукції через мережу Інтернет, результати аналізу вебсайтів Florium, Dicentra та DonPion, а також функціональні можливості, які планується реалізувати в межах проєкту. У результаті було сформовано функціональні та нефункціональні вимоги до інформаційної системи.

Функціональні вимоги описують можливості, які повинні бути доступними користувачам та адміністратору під час роботи із системою. До них належать перегляд каталогу продукції, пошук товарів, оформлення замовлень, робота з особистим кабінетом та адміністрування інформації про товари. Основні функціональні вимоги до розроблюваної системи наведено в таблиці 1.2.

Таблиця 1.2

Функціональні вимоги до інформаційної системи

№	Функціональна вимога	Опис
1.	Перегляд каталогу товарів	Система повинна забезпечувати перегляд усіх товарів, доступних для замовлення
2.	Робота з категоріями	Користувач повинен мати можливість переглядати товари за категоріями
3.	Пошук товарів	Система повинна підтримувати пошук продукції за назвою
4.	Фільтрація товарів	Користувач повинен мати можливість відбирати товари за визначеними параметрами
5.	Перегляд сторінки товару	Для кожного товару повинна відображатися детальна інформація, фотографії та вартість
6.	Робота з кошиком	Система повинна забезпечувати додавання, видалення та зміну кількості товарів у кошику
7.	Оформлення замовлення	Користувач повинен мати можливість оформити замовлення через вебінтерфейс
8.	Реєстрація та авторизація	Система повинна підтримувати створення облікового запису та вхід користувача
9.	Особистий кабінет	Зареєстрований користувач повинен мати доступ до особистої інформації
10.	Індивідуальне замовлення	Система повинна надавати можливість створення індивідуального запиту на формування букета
11.	Адміністрування товарів	Адміністратор повинен мати можливість додавати, редагувати та видаляти товари
12.	Керування замовленнями	Адміністратор повинен мати можливість переглядати та змінювати статуси замовлень

Функціональні можливості, наведені в таблиці 1.2, охоплюють основні сценарії роботи із системою. Вони забезпечують взаємодію користувача з вебсайтом на всіх етапах — від перегляду каталогу продукції до оформлення замовлення. Окремо передбачено можливості для адміністратора, які дозволяють керувати товарами та контролювати обробку замовлень.

Окрім реалізації необхідного функціоналу, вебсайт повинен відповідати вимогам щодо зручності використання, продуктивності та надійності роботи. Такі характеристики належать до нефункціональних вимог і визначають якість роботи програмного забезпечення незалежно від конкретних функцій системи. Основні нефункціональні вимоги наведено в таблиці 1.3.

Таблиця 1.3

Нефункціональні вимоги до інформаційної системи

№	Нефункціональна вимога	Опис
1.	Адаптивність	Коректна робота вебсайту на персональних комп'ютерах, планшетах та смартфонах
2.	Зручність використання	Інтерфейс повинен бути зрозумілим та забезпечувати швидкий доступ до основних функцій
3.	Продуктивність	Сторінки повинні завантажуватися без помітних затримок для користувача
4.	Надійність	Система повинна стабільно працювати під час виконання основних операцій
5.	Масштабованість	Архітектура повинна дозволяти розширення функціональних можливостей у майбутньому
6.	Безпека даних	Дані користувачів повинні зберігатися та оброблятися відповідно до вимог безпеки
7.	Підтримуваність	Структура програмного коду повинна забезпечувати можливість подальшого супроводу та розвитку системи

Нефункціональні вимоги визначають умови, за яких вебсайт повинен забезпечувати стабільну та комфортну роботу користувачів. Для розроблюваної системи особливе значення мають адаптивність інтерфейсу, швидке завантаження сторінок і можливість подальшого розширення функціоналу без суттєвої зміни архітектури проєкту.

Сформований перелік функціональних і нефункціональних вимог використовується як основа для подальшого проєктування вебсайту BloomShop. На наступному етапі доцільно розглянути сучасні архітектурні підходи та технології веброзробки, які можуть бути використані для реалізації визначених вимог.

1.3 Порівняльний аналіз архітектурних рішень та технологій розробки

Під час розробки вебсайту комерційної реалізації квітів необхідно обрати технології, які забезпечать створення інтерактивного інтерфейсу, зручну організацію програмного коду та можливість подальшого розвитку проєкту. Для реалізації системи було розглянуто сучасні засоби розробки клієнтської та

серверної частин вебдодатків, які найчастіше використовуються під час створення інформаційних систем електронної комерції [18, 26].

Для реалізації клієнтської частини були проаналізовані React, Vue та Angular. Усі зазначені технології підтримують створення односторінкових вебдодатків (SPA), компонентний підхід до побудови інтерфейсу та роботу з динамічними даними.

React є однією з найбільш поширених бібліотек для створення користувацьких інтерфейсів [18, 26]. Основою React є компонентний підхід, за якого інтерфейс складається з окремих незалежних елементів. Така організація проєкту дозволяє повторно використовувати програмний код, спрощує підтримку вебдодатку та полегшує розширення функціональних можливостей. Для розробки вебсайту BloomShop цей підхід є доцільним, оскільки більшість сторінок містять однакові елементи інтерфейсу, зокрема картки товарів, меню навігації, форми та модальні вікна.

Vue також підтримує компонентну структуру проєкту та дозволяє створювати сучасні вебінтерфейси. Його перевагою є відносно проста структура та швидке освоєння. Разом із тим для реалізації більш складних проєктів часто використовується менша кількість готових рішень і додаткових бібліотек порівняно з React.

Angular є повноцінним фреймворком із великою кількістю вбудованих інструментів для розробки вебдодатків. Він підтримує маршрутизацію, роботу з даними та організацію великих проєктів. Водночас Angular має складнішу структуру та потребує більшого обсягу програмного коду. Для вебсайту середнього рівня складності використання такого підходу може бути надлишковим [5].

Проведений аналіз показав, що всі розглянуті рішення дозволяють створювати сучасні вебдодатки. Проте React поєднує компонентний підхід, гнучкість розробки та широку екосистему додаткових бібліотек, що робить його придатним для реалізації інтерактивного вебсайту електронної комерції.

Для створення серверної частини були розглянуті Flask, Django та Node.js [32, 39]. Дані технології використовуються для реалізації бізнес-логіки застосунків, взаємодії з базами даних та обробки запитів від клієнтської частини.

Flask є легким вебфреймворком для мови Python, який дозволяє будувати серверні застосунки без надлишкових компонентів [32]. Його структура дає можливість самостійно визначати організацію проєкту та підключати лише ті модулі, які необхідні для реалізації конкретного функціоналу. Для вебсайту BloomShop такий підхід є зручним завдяки відносно невеликій кількості серверної логіки та використанню REST API для взаємодії з клієнтською частиною.

Django також належить до популярних Python-фреймворків. На відміну від Flask, він містить значну кількість готових компонентів, серед яких система адміністрування, механізми автентифікації та засоби роботи з базами даних. Для великих проєктів це є перевагою, проте частина функціоналу може залишатися невикористаною, що ускладнює структуру невеликих інформаційних систем.

Node.js дозволяє реалізовувати серверну частину із використанням JavaScript. Перевагою такого підходу є використання однієї мови програмування як на стороні клієнта, так і на стороні сервера. Водночас для організації повноцінного серверного застосунку необхідно додатково обирати та налаштовувати набір бібліотек, що збільшує складність проєкту.

Порівняння розглянутих технологій показало, що кожне рішення має власні переваги та може використовуватися для створення вебдодатків електронної комерції. Для розроблюваної системи важливими є простота реалізації, можливість подальшого розширення функціоналу та зручність підтримки програмного коду. З урахуванням цих критеріїв доцільно використовувати React для клієнтської частини та Flask для серверної частини вебсайту. Остаточне обґрунтування вибору технологічного стеку наведено в наступному підрозділі.

1.4 Обґрунтування вибору технологічного стеку

Після аналізу сучасних технологій веброзробки було обрано набір інструментів, який найбільше відповідає вимогам розроблюваного вебсайту комерційної реалізації квітів. Під час вибору враховувалися особливості проєкту, необхідність реалізації інтерактивного користувацького інтерфейсу, підтримка клієнт-серверної архітектури та можливість подальшого розширення функціоналу системи.

Для реалізації клієнтської частини вебсайту обрано бібліотеку React [18, 26]. Дане рішення дозволяє будувати інтерфейс із незалежних компонентів, що спрощує структуру проєкту та повторне використання програмного коду. У межах вебсайту BloomShop компонентний підхід використовується під час реалізації каталогу товарів, карток продукції, сторінок товарів, кошика, форм оформлення замовлень та інших елементів інтерфейсу.

Організація навігації між сторінками реалізується за допомогою React Router [27]. Використання даної бібліотеки дозволяє здійснювати переходи між сторінками без повного перезавантаження вебдодатку. У проєкті маршрутизація використовується для переходу між головною сторінкою, каталогом, сторінками окремих товарів, оформленням замовлення та адміністративною частиною системи.

Для стилізації інтерфейсу використовується Tailwind CSS [28]. Даний фреймворк надає набір готових утилітарних класів, що дозволяє швидко створювати адаптивний дизайн та забезпечувати коректне відображення сторінок на різних типах пристроїв. Використання Tailwind CSS також спрощує підтримку стилів і зменшує кількість окремих CSS-файлів у проєкті.

Серверна частина реалізується за допомогою Flask [32]. Вибір даного фреймворку обумовлений його простою структурою та можливістю створення REST API без використання надлишкових компонентів. У межах проєкту Flask використовується для обробки запитів від клієнтської частини, роботи з базою даних та реалізації серверної логіки вебсайту.

Для зберігання інформації про товари, користувачів, замовлення та індивідуальні заявки використовується SQLite [34]. Дана система керування базами даних не потребує встановлення окремого сервера та є достатньою для реалізації вебпроєкту даного масштабу.

Взаємодія із базою даних реалізована за допомогою SQLAlchemy [33]. Використання ORM дозволяє працювати з даними через об'єкти Python, що спрощує розробку серверної частини та підвищує зручність підтримки програмного коду.

Для обміну даними між клієнтською та серверною частинами використовується архітектурний стиль REST [38, 39]. Передавання інформації здійснюється у форматі JSON, що забезпечує зручну взаємодію між React-застосунком та серверною частиною на Flask.

Обраний набір технологій відповідає функціональним і нефункціональним вимогам, визначеним у попередньому підрозділі. Поєднання React, Flask та SQLite дозволяє реалізувати вебсайт із сучасним користувацьким інтерфейсом, зручною структурою навігації та можливістю подальшого розвитку системи.

Таблиця 1.4

Технологічний стек проєкту

Компонент системи	Технологія	Призначення
Frontend	React	Реалізація користувацького інтерфейсу
Маршрутизація	React Router	Навігація між сторінками
Стилізація	Tailwind CSS	Оформлення та адаптивний дизайн
Backend	Flask	Серверна логіка та API
ORM	SQLAlchemy	Робота з базою даних
База даних	SQLite	Зберігання інформації
Обмін даними	REST API	JSON Взаємодія між клієнтом і сервером

Наведений у таблиці 1.4 технологічний стек охоплює всі основні компоненти вебсайту BloomShop. Обрані інструменти забезпечують реалізацію клієнтської та серверної частин системи, підтримують роботу з базою даних і дозволяють організувати взаємодію між компонентами через REST API.

Проведений аналіз показав, що використання React, Flask, SQLite та SQLAlchemy є достатнім для реалізації функціональних можливостей, визначених у попередніх підрозділах. Обрані технології дозволяють створити вебсайт комерційної реалізації квітів із підтримкою каталогу товарів, кошика, оформлення замовлень, особистого кабінету користувача та адміністративної панелі.

Висновки до розділу 1

У першому розділі розглянуто особливості реалізації квіткової продукції через мережу Інтернет та проаналізовано сучасні вебресурси, що працюють у даній сфері. Під час аналізу вебсайтів Florium, Dicentra та DonPion було визначено підходи до організації каталогів товарів, навігації, пошуку продукції та оформлення замовлень. Результати аналізу дозволили виділити функціональні рішення, які доцільно використати під час розробки вебсайту BloomShop, а також виявити окремі недоліки існуючих систем.

На основі дослідження предметної області сформовано функціональні та нефункціональні вимоги до розроблюваної інформаційної системи. Визначено основні можливості вебсайту, серед яких перегляд каталогу продукції, пошук товарів, робота з кошиком, оформлення замовлень, особистий кабінет користувача та засоби адміністрування. Окрему увагу приділено вимогам щодо адаптивності інтерфейсу, зручності використання та стабільності роботи системи.

У межах розділу також виконано порівняння сучасних технологій веброзробки для створення клієнтської та серверної частин вебдодатку. За результатами аналізу обґрунтовано вибір технологічного стеку, до складу якого входять React, React Router, Tailwind CSS, Flask, SQLAlchemy та SQLite. Обрані інструменти відповідають вимогам проєкту та дозволяють реалізувати вебсайт із клієнт-серверною архітектурою, підтримкою REST API та можливістю подальшого розширення функціоналу.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ КОМЕРЦІЙНОЇ РЕАЛІЗАЦІЇ КВІТІВ

2.1 Проєктування структури вебсайту

На етапі проєктування було визначено структуру вебсайту BloomShop, перелік основних сторінок та логіку переходів між ними. Побудова структури виконувалася з урахуванням особливостей інтернет-магазину квіткової продукції та функціональних вимог, сформованих у першому розділі.

Під час проєктування передбачалося скорочення кількості дій, необхідних для переходу від перегляду товарів до оформлення замовлення. Окремо реалізовано сторінки оформлення замовлення, особистого кабінету користувача та адміністративної панелі.

Центральним елементом структури є головна сторінка, з якої користувач отримує доступ до каталогу товарів, кошика, оформлення замовлення, особистого кабінету та інших розділів системи. Каталог поділено на п’ять категорій: букети, квіти в коробках, монобукети, рослини та подарунки. Такий поділ відповідає структурі асортименту, реалізованій у вебсайті.

Для кожного товару передбачено окрему сторінку, на якій відображаються фотографії, опис, склад композиції, вартість та можливість додавання товару до кошика. У каталозі реалізовано пошук товарів за назвою та перехід між категоріями продукції.

Для зареєстрованих користувачів передбачено особистий кабінет, а для керування товарами та замовленнями — адміністративну панель. Адміністратор має можливість додавати, редагувати та видаляти товари, а також переглядати інформацію про замовлення користувачів.

Структуру навігації розробленого вебсайту наведено на рисунку 2.1.

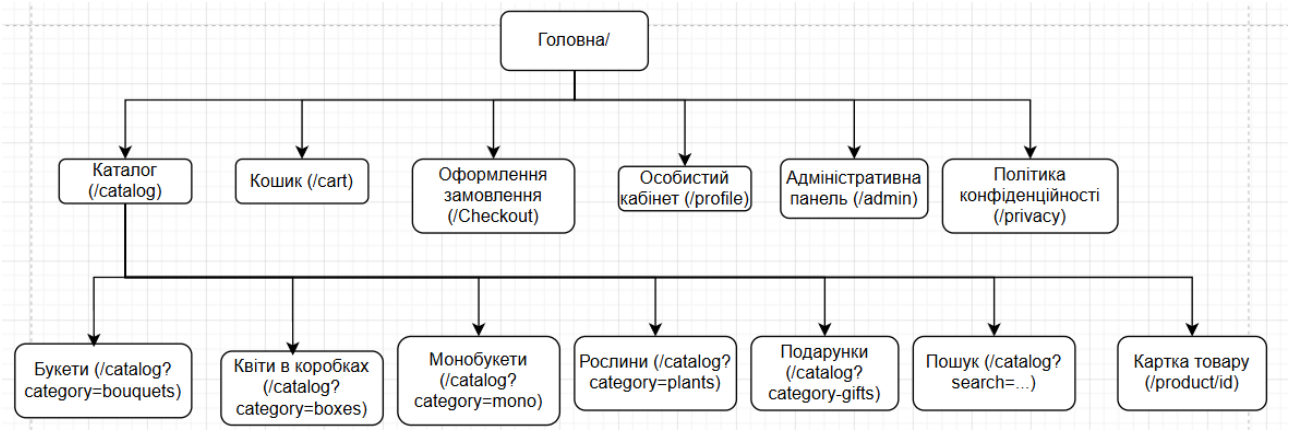


Рисунок 2.1. Схема структури навігації вебсайту

2.1.1 Аналіз структури користувацького інтерфейсу

Структура навігації, наведена на рисунку 2.1, відображає основні сторінки вебсайту BloomShop та взаємозв'язки між ними. Під час проєктування інтерфейсу основна увага приділялася забезпеченню швидкого доступу до каталогу товарів, оформлення замовлення та інших функцій системи [3, 4].

Початковою сторінкою для користувача є головна сторінка вебсайту. На ній розміщено банер, блок категорій товарів, популярні позиції каталогу, переваги сервісу, відгуки клієнтів та форму індивідуального замовлення букета. Така структура дозволяє ознайомитися з основними можливостями вебсайту та перейти до перегляду товарів.

Основним розділом системи є каталог продукції. У каталозі представлено товари, згруповані за категоріями: букети, квіти в коробках, монобукети, рослини та подарунки. Для спрощення пошуку необхідної продукції реалізовано пошук товарів за назвою та можливість перегляду окремих категорій.

Для кожного товару передбачено окрему сторінку, на якій відображаються фотографії, опис, склад композиції, вартість, рейтинг та інша інформація. Також користувач може обрати кількість товару, розмір композиції та додати товар до кошика.

Після вибору продукції користувач переходить до кошика, де відображаються додані товари, їх кількість та загальна вартість замовлення. Передбачено можливість зміни кількості товарів або видалення окремих позицій зі списку замовлення.

Наступним етапом є оформлення замовлення. На відповідній сторінці користувач заповнює контактні дані, адресу доставки та додаткову інформацію, необхідну для обробки замовлення.

Для зареєстрованих користувачів передбачено особистий кабінет, який забезпечує доступ до інформації профілю та персональних функцій системи. Для керування вмістом вебсайту використовується адміністративна панель, через яку здійснюється робота з товарами та замовленнями.

Проведений аналіз структури інтерфейсу показав, що розроблена система охоплює всі основні етапи взаємодії користувача з вебсайтом — від перегляду каталогу до оформлення замовлення. Побудова інтерфейсу на основі логічно пов’язаних сторінок дозволяє спростити навігацію та забезпечити зручну роботу з вебсайтом на різних типах пристроїв.

2.1.2 Проєктування навігації між сторінками

Під час проєктування вебсайту BloomShop було визначено основні маршрути переходу між сторінками та порядок взаємодії користувача з функціональними модулями системи. Навігація побудована таким чином, щоб користувач міг швидко перейти від перегляду товарів до оформлення замовлення без зайвих переходів між сторінками [3, 4].

Основним навігаційним елементом вебсайту є верхнє меню, яке відображається на всіх сторінках системи. Через нього користувач отримує доступ до каталогу продукції, кошика, особистого кабінету та інших розділів вебсайту. Також у шапці відображаються засоби пошуку товарів і швидкого переходу до кошика.

Типовий сценарій роботи починається з головної сторінки, де користувач може переглянути популярні товари, категорії продукції та інші інформаційні

блоки. Після цього здійснюється перехід до каталогу товарів, у якому доступний перегляд асортименту за категоріями та пошук продукції за назвою.

З каталогу користувач переходить на сторінку окремого товару, де відображаються фотографії, опис, склад композиції, вартість та додаткові характеристики. На цій сторінці також реалізовано додавання товару до кошика та вибір параметрів замовлення.

Після формування списку товарів користувач переходить до кошика, де може переглянути обрані позиції, змінити їх кількість або видалити окремі товари. Наступним етапом є сторінка оформлення замовлення, на якій вводяться контактні дані та адреса доставки.

Для авторизованих користувачів передбачено доступ до особистого кабінету. Окремо реалізовано адміністративну панель, яка використовується для керування товарами та роботи із замовленнями. Доступ до адміністративної частини системи надається лише адміністратору.

Під час проєктування навігації також враховувалися вимоги адаптивності інтерфейсу. Усі основні розділи вебсайту залишаються доступними як на персональних комп’ютерах, так і на мобільних пристроях.

Розроблена структура навігації забезпечує послідовний перехід між сторінками системи та підтримує основні сценарії взаємодії користувача з вебсайтом — від перегляду продукції до оформлення замовлення.

2.1.3 Проєктування структури каталогу товарів

Каталог товарів є основним розділом вебсайту BloomShop, через який користувач отримує доступ до асортименту продукції та здійснює пошук необхідних товарів. Під час проєктування каталогу основна увага приділялася зручності навігації, швидкому доступу до товарів та зрозумілому відображенню інформації.

Структура каталогу формувалася з урахуванням особливостей квіткової продукції та асортименту, представленого у вебсайті. Для впорядкування товарів

було використано поділ на окремі категорії, що дозволяє користувачу швидко перейти до потрібного типу продукції без перегляду всього каталогу.

У межах проєкту передбачено п’ять категорій товарів:

- букети;
- квіти в коробках;
- монобукети;
- рослини;
- подарунки.

Кожна категорія містить окремий набір товарів та відображається у каталозі через систему фільтрації. Такий підхід спрощує перегляд продукції та дозволяє швидко знаходити потрібний розділ каталогу.

Для демонстрації роботи вебсайту було сформовано асортимент із сорока товарів, рівномірно розподілених між усіма категоріями. Для кожного товару передбачено назву, опис, вартість, рейтинг та набір фотографій, які використовуються на сторінці каталогу та сторінці окремого товару.

У каталозі товари відображаються у вигляді карток. Картка містить фотографію продукції, назву, ціну, рейтинг та додаткові елементи інтерфейсу. Після вибору товару користувач переходить на сторінку детального перегляду, де може ознайомитися з повною інформацією про продукцію та додати її до кошика.

Для спрощення роботи з великим асортиментом реалізовано пошук товарів за назвою та систему сортування продукції. Це дозволяє швидко знаходити необхідні позиції незалежно від їх категорії та місця розташування в каталозі.

Під час проєктування каталогу також враховувалися вимоги адаптивності інтерфейсу. Кількість карток у рядку автоматично змінюється залежно від розміру екрана пристрою, що забезпечує коректне відображення каталогу на персональних комп’ютерах, планшетах та смартфонах.

Структура каталогу побудована таким чином, щоб у майбутньому можна було додавати нові товари та категорії без зміни загальної логіки роботи системи.

Завдяки цьому каталог може розширюватися разом із розвитком вебсайту та збільшенням асортименту продукції.

Спроєктована структура каталогу забезпечує впорядковане відображення товарів, підтримує пошук і фільтрацію продукції та відповідає функціональним вимогам, визначеним для вебсайту комерційної реалізації квітів.

2.2 Проєктування frontend-архітектури вебдодатку

Frontend-частина вебдодатку забезпечує взаємодію користувача із системою та відповідає за відображення інформації, навігацію між сторінками й обробку дій користувача. Під час проєктування клієнтської частини вебсайту BloomShop було обрано архітектурний підхід, який забезпечує зручну підтримку програмного коду, повторне використання компонентів та можливість подальшого розширення функціональності системи.

Для реалізації frontend-частини використано бібліотеку React, яка підтримує компонентний підхід до розробки користувацьких інтерфейсів та широко застосовується під час створення сучасних вебдодатків [16, 18, 26]. Використання React дозволяє будувати інтерфейс із незалежних компонентів, кожен із яких відповідає за окрему частину функціоналу вебсайту.

Під час проєктування архітектури особлива увага приділялася повторному використанню компонентів. У проєкті окремими компонентами реалізовано шапку сайту, підвал, картки товарів, блоки головної сторінки, форми введення даних та інші елементи інтерфейсу. Такий підхід дозволяє уникнути дублювання програмного коду та спрощує подальше внесення змін до системи.

Архітектура клієнтської частини побудована за принципом односторінкового вебдодатку (Single Page Application, SPA), коли оновлення вмісту сторінок відбувається без повного перезавантаження браузера [16, 26]. Це забезпечує швидшу роботу інтерфейсу та покращує зручність використання вебсайту.

Для реалізації навігації між сторінками використовується бібліотека React Router [27]. Вона забезпечує маршрутизацію між основними розділами системи,

зокрема головною сторінкою, каталогом товарів, сторінкою окремого товару, кошиком та сторінкою оформлення замовлення. Використання маршрутизації дозволяє організувати логічну структуру вебдодатку та спростити навігацію для користувача.

Створення та збірка проєкту виконувалися за допомогою середовища Vite [29], яке забезпечує швидкий запуск застосунку під час розробки та підтримує сучасні інструменти frontend-розробки.

Для стилізації інтерфейсу використовується фреймворк Tailwind CSS [28]. Його застосування дозволило реалізувати адаптивний дизайн вебсайту та забезпечити коректне відображення сторінок на персональних комп'ютерах, планшетах і смартфонах без створення великої кількості окремих CSS-файлів.

Таким чином, обрана frontend-архітектура поєднує компонентний підхід React, клієнтську маршрутизацію React Router та засоби адаптивної стилізації Tailwind CSS. Таке поєднання технологій відповідає вимогам проєкту та забезпечує зручну основу для реалізації функціональних модулів вебсайту BloomShop.

2.2.1 Компонентна структура React-додатку

Під час розробки клієнтської частини вебсайту BloomShop використано компонентний підхід, який є одним із базових принципів побудови застосунків на основі React [16, 18, 26]. Застосування такого підходу дозволяє розділити інтерфейс на окремі функціональні елементи, кожен із яких виконує власне завдання та може використовуватися в різних частинах системи.

Структура проєкту організована таким чином, щоб логічно розділити сторінки, компоненти та службові модулі. Основні елементи інтерфейсу розміщені в каталозі components, де виділено окремі групи компонентів для головної сторінки, елементів навігації та багаторазового використання.

До базових компонентів належать Header та Footer, які відображаються на більшості сторінок вебсайту та забезпечують доступ до основних розділів

системи. Через Header реалізовано навігацію між сторінками, пошук товарів, перехід до кошика та особистого кабінету користувача.

Для формування головної сторінки використовуються окремі компоненти Hero, Categories, Benefits, PopularProducts, Reviews, WowSection та CustomOrder. Кожен із них відповідає за відображення певного інформаційного блоку та може редагуватися незалежно від інших частин інтерфейсу.

Одним із ключових компонентів системи є ProductCard, який використовується для відображення товарів у каталозі, на головній сторінці та в рекомендаціях. Картка містить фотографію товару, назву, рейтинг, вартість та елементи взаємодії з користувачем. Використання єдиного компонента забезпечує однаковий стиль відображення продукції в межах усього вебсайту.

Окремими сторінковими компонентами реалізовано каталог товарів, сторінку окремого товару, оформлення замовлення та адміністративну панель. Для роботи з даними користувача і кошиком використовуються контексти AuthContext та CartContext, які забезпечують централізоване керування станом застосунку та обмін даними між компонентами [16, 26].

Для реалізації індивідуального підбору букетів використовується окремий модуль CustomOrder, який дозволяє обрати квіти, вказати бюджет та сформувати заявку на створення композиції за індивідуальними побажаннями. Додатково реалізовано панель профілю користувача ProfilePanel, через яку здійснюється доступ до персональних функцій системи.

Використання компонентної структури дозволило організувати клієнтську частину вебсайту у вигляді взаємопов’язаних модулів із чітким розподілом відповідальності між ними. Такий підхід спрощує підтримку програмного коду, полегшує розширення функціональності та відповідає сучасним рекомендаціям щодо розробки React-застосунків [16, 18, 26].

2.2.2 Організація структури директорій проєкту

Під час розробки вебсайту BloomShop було сформовано структуру директорій, яка забезпечує розділення сторінок, компонентів, даних та

допоміжних ресурсів. Такий підхід спрощує підтримку програмного коду та дозволяє швидко знаходити необхідні файли під час подальшого розвитку проєкту.

Основна частина клієнтського застосунку розміщується в директорії `src`, де зосереджено компоненти інтерфейсу, сторінки вебсайту, контексти, локальні дані та графічні матеріали. Центральними файлами клієнтської частини є `main.jsx` та `App.jsx`, які відповідають за запуск застосунку та налаштування маршрутизації між сторінками [26, 27].

Для реалізації інтерфейсу використовується директорія `components`, у якій компоненти згруповані за призначенням. Зокрема, у піддиректорії `layout` розміщені компоненти Header та Footer, що відповідають за навігацію та відображаються на більшості сторінок вебсайту. У директорії `home` знаходяться компоненти головної сторінки, серед яких Hero, Categories, Benefits, PopularProducts, Reviews та інші інформаційні блоки. Для багаторазово використовуваних елементів інтерфейсу передбачено директорію `common`, де розміщено компонент ProductCard.

Сторінкові компоненти винесено до директорії `pages`. У ній розташовані сторінки каталогу товарів, сторінка окремого товару, оформлення замовлення та інші модулі вебсайту. Такий поділ дозволяє відокремити логіку сторінок від допоміжних компонентів інтерфейсу та спрощує навігацію в структурі проєкту.

Для роботи з даними на етапі розробки використовується директорія `data`, де зберігаються файли з описом товарів і категорій. У масиві товарів для кожної позиції визначаються назва, категорія, вартість, рейтинг, опис, склад композиції та шляхи до зображень. Це дозволяє тестувати роботу каталогу без постійного звернення до серверної частини.

Графічні матеріали розміщені в директорії `assets`. Зображення товарів згруповані за категоріями продукції: букети, квіти в коробках, монобукети, рослини та подарунки. Окремо зберігаються банери, іконки та інші елементи оформлення інтерфейсу.

Для централізованого керування станом застосунку використовується директорія `context`, у якій розміщено файли `CartContext` та `AuthContext`. Вони забезпечують роботу кошика, авторизації користувача та передачу даних між компонентами без дублювання логіки [16, 26].

Узагальнена структура frontend-частини проєкту має такий вигляд:

• src/ — основна директорія клієнтського застосунку;

• components/ — компоненти інтерфейсу;

• pages/ — сторінки вебсайту;

• data/ — локальні дані про товари та категорії;

• assets/ — зображення та графічні ресурси;

• context/ — керування глобальним станом застосунку;

• App.jsx — маршрутизація сторінок;

• main.jsx — запуск клієнтської частини.

Запропонована структура директорій відповідає компонентному підходу React [16, 18, 26] та дозволяє підтримувати зрозумілу організацію проєкту навіть після додавання нових сторінок, компонентів або функціональних модулів..

2.2.3 Реалізація клієнтської маршрутизації

Для організації переходів між сторінками вебсайту BloomShop використовується клієнтська маршрутизація, реалізована за допомогою бібліотеки React Router DOM [27]. Застосування такого підходу дозволяє побудувати односторінковий вебдодаток (SPA), у якому зміна сторінок відбувається без повного перезавантаження браузера [16, 26].

Клієнтська маршрутизація забезпечує швидке переміщення між розділами вебсайту та дозволяє користувачу працювати з каталогом, товарами й замовленнями без додаткових затримок. Усі маршрути налаштовуються в основному файлі застосунку та пов’язуються з відповідними сторінковими компонентами.

Початковим маршрутом системи є головна сторінка, яка відкривається за адресою /. На ній розміщено основні інформаційні блоки вебсайту, категорії продукції, популярні товари та форму індивідуального замовлення букета.

Для перегляду асортименту використовується маршрут /catalog, який забезпечує доступ до каталогу товарів. У межах каталогу реалізовано пошук продукції, перегляд категорій та сортування товарів за заданими параметрами.

Перехід до детальної інформації про продукцію виконується через динамічний маршрут /product/:id. Такий підхід дозволяє використовувати один сторінковий компонент для відображення інформації про будь-який товар із каталогу. Ідентифікатор товару передається через адресу сторінки та використовується для отримання відповідних даних.

Для завершення процесу покупки використовується маршрут /checkout, який відкриває сторінку оформлення замовлення. На цій сторінці користувач вводить контактні дані та інформацію, необхідну для доставки продукції.

Окремо в системі реалізовано функції авторизації користувача, роботи з кошиком та адміністративними елементами керування. Доступ до них здійснюється через відповідні компоненти інтерфейсу та механізми маршрутизації, інтегровані у структуру вебдодатку.

Основні маршрути клієнтської частини наведено в таблиці 2.1.

Таблиця 2.1

Основні маршрути вебдодатку

Маршрут	Призначення
/	Головна сторінка
/catalog	Каталог товарів
/product/	Сторінка окремого товару
/cart	Кошик
/checkout	Оформлення замовлення

Використання React Router DOM дозволило реалізувати логічну структуру переходів між сторінками вебсайту та забезпечити зручну навігацію між основними функціональними модулями системи [27].

2.3 Проєктування backend-архітектури та бази даних

Після проєктування клієнтської частини вебсайту було визначено структуру серверної частини та механізми зберігання даних. Backend-компоненти забезпечують обробку запитів від користувачів, взаємодію з базою даних та передачу інформації до frontend-частини вебдодатку [6, 12].

У розробленому вебсайті серверна частина реалізується за допомогою фреймворку Flask, який використовується для створення REST API та організації взаємодії між клієнтом і сервером [32]. Через API клієнтська частина отримує інформацію про товари, надсилає дані замовлень та виконує інші операції, необхідні для роботи системи.

Під час проєктування було обрано клієнт-серверну архітектуру, у якій frontend та backend функціонують як окремі компоненти системи. Такий підхід дозволяє незалежно розвивати інтерфейс користувача та серверну логіку, а також спрощує подальше розширення функціональних можливостей вебсайту [12, 38].

Для зберігання інформації використовується реляційна база даних SQLite, яка містить дані про товари, користувачів, замовлення та інші об’єкти системи [34, 35]. Робота з базою даних виконується через ORM-бібліотеку SQLAlchemy, що дозволяє здійснювати доступ до даних за допомогою об’єктів Python без написання великої кількості SQL-запитів [33].

Передача інформації між клієнтською та серверною частинами здійснюється відповідно до принципів REST API із використанням формату JSON [38, 39]. Такий підхід забезпечує стандартизований обмін даними між компонентами системи та спрощує інтеграцію frontend і backend частин вебдодатку.

У межах даного підрозділу розглядаються архітектура взаємодії клієнтської та серверної частин, структура бази даних і програмний інтерфейс взаємодії, який використовується у вебсайті BloomShop.

2.3.1 Архітектура взаємодії frontend та backend частин системи

Для реалізації інформаційної системи комерційної реалізації квітів використано клієнт-серверну архітектуру, яка передбачає розподіл функціональності між клієнтською та серверною частинами. Такий підхід дозволяє відокремити користувацький інтерфейс від бізнес-логіки та механізмів зберігання даних, що спрощує супровід і подальший розвиток програмного забезпечення [12, 38].

Клієнтська частина реалізована за допомогою бібліотеки React і відповідає за відображення інформації, взаємодію з користувачем та формування запитів до серверної частини системи [16, 18, 26]. Через інтерфейс користувач отримує доступ до каталогу товарів, сторінок продукції, кошика, оформлення замовлення, особистого кабінету та інших функціональних модулів вебсайту.

Серверна частина побудована на основі мікрофреймворку Flask. Її основними завданнями є обробка HTTP-запитів, реалізація бізнес-логіки застосунку та взаємодія з базою даних [31, 32]. Для обміну інформацією між клієнтською та серверною частинами використовується REST API, а передавання даних здійснюється у форматі JSON [38, 39].

Для роботи з базою даних використовується ORM-бібліотека SQLAlchemy, яка забезпечує взаємодію між серверною логікою та реляційною базою даних SQLite [23, 33, 34]. Такий підхід дозволяє виконувати операції створення, отримання, оновлення та видалення даних без формування великої кількості SQL-запитів вручну.

Загальну архітектуру взаємодії компонентів системи наведено на рисунку 2.2.

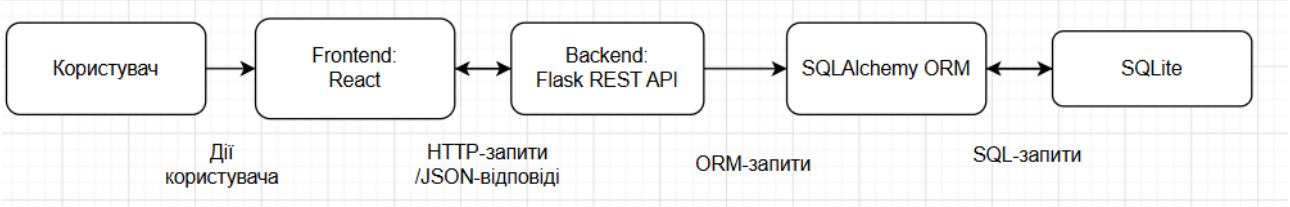


Рисунок 2.2. Архітектура взаємодії frontend та backend

Як показано на рисунку 2.2, взаємодія із системою починається з виконання користувачем певної дії у вебінтерфейсі. Після цього клієнтська частина формує HTTP-запит до серверної частини. Сервер обробляє отримані дані, виконує необхідні операції та за потреби звертається до бази даних через SQLAlchemy ORM.

У базі даних зберігається інформація про товари, користувачів, замовлення та інші сутності системи. Після виконання необхідних операцій сервер формує JSON-відповідь та повертає її клієнтській частині, яка оновлює відображення інформації без повного перезавантаження сторінки.

Запропонована архітектура використовується під час перегляду каталогу товарів, відкриття сторінок продукції, роботи з кошиком, оформлення замовлень, авторизації користувачів та виконання адміністративних операцій. Такий підхід забезпечує розмежування відповідальності між окремими рівнями системи, спрощує її масштабування та відповідає сучасним принципам розробки вебдодатків електронної комерції [12, 38].

2.3.2 Проєктування структури бази даних

Для роботи вебдодатку необхідно організувати зберігання інформації про товари, користувачів, замовлення та індивідуальні заявки. З цією метою під час проєктування системи було розроблено структуру реляційної бази даних, яка використовується для збереження та обробки даних, необхідних для функціонування вебсайту [14, 15, 35].

Під час створення структури бази даних враховувалися функціональні вимоги, сформовані на етапі аналізу предметної області. Зокрема, необхідно було реалізувати зберігання інформації про асортимент продукції, категорії товарів, облікові записи користувачів, оформлені замовлення та заявки на створення індивідуальних квіткових композицій [13, 14].

Для реалізації проєкту обрано систему керування базами даних SQLite. Дане рішення не потребує окремого сервера баз даних, легко інтегрується з Flask та SQLAlchemy і є достатнім для реалізації вебдодатку такого масштабу [34].

Структура бази даних складається з таблиць categories, products, users, orders, order_items та custom_orders. Кожна таблиця відповідає за зберігання окремого типу інформації та використовується під час роботи відповідних модулів системи.

Структуру бази даних вебдодатку наведено на рисунку 2.3.

Рисунок 2.3. Структура бази даних вебсайту

Як показано на рисунку 2.3, таблиця categories використовується для зберігання категорій товарів. Для кожної категорії визначаються назва, службовий ідентифікатор та допоміжні поля. З таблицею категорій пов’язана таблиця products, у якій міститься інформація про продукцію, що відображається в каталозі.

У таблиці products зберігаються назва товару, категорія, вартість, рейтинг, опис, фотографії, склад композиції та додаткова інформація щодо доставки. Дані

цієї таблиці використовуються під час формування каталогу та сторінок окремих товарів.

Інформація про зареєстрованих користувачів зберігається в таблиці users. Для кожного користувача фіксуються ім’я, адреса електронної пошти, пароль у зашифрованому вигляді та службові відомості. Дані цієї таблиці використовуються для авторизації та роботи особистого кабінету користувача [32].

Для зберігання оформлених замовлень використовується таблиця orders. У ній містяться контактні дані замовника, адреса доставки, спосіб оплати, статус замовлення та його загальна вартість. Кожне замовлення пов’язується з відповідним користувачем через поле user_id.

Оскільки одне замовлення може містити декілька товарів, у структурі бази даних передбачено таблицю order_items. У ній зберігаються окремі позиції замовлення, їх кількість, вартість та обраний розмір композиції. Через цю таблицю реалізовано зв’язок між замовленнями та товарами.

Для збереження індивідуальних заявок на створення букетів використовується таблиця custom_orders. У ній містяться контактні дані клієнта, перелік бажаних квітів, бюджет, дата доставки та додаткові побажання щодо майбутньої композиції.

У структурі бази даних передбачено зв’язки між категоріями та товарами, користувачами і замовленнями, а також між замовленнями та їх позиціями. Один користувач може оформити декілька замовлень, а кожне замовлення може містити декілька товарів. Така структура спрощує роботу з даними та дозволяє зберігати інформацію без дублювання [35, 37].

Розроблена структура бази даних використовується для зберігання інформації про товари, користувачів, замовлення та індивідуальні заявки. Вона забезпечує роботу каталогу продукції, механізмів оформлення замовлень, авторизації користувачів та обробки індивідуальних запитів на створення квіткових композицій.

2.3.3 Проєктування програмного інтерфейсу взаємодії (REST API)

Для обміну даними між клієнтською та серверною частинами вебдодатку використовується архітектурний стиль REST API. Такий підхід дозволяє організувати взаємодію між компонентами системи через набір HTTP-запитів і забезпечує незалежність frontend та backend частин застосунку [38, 39].

Під час роботи вебсайту клієнтська частина надсилає запити до сервера для отримання або збереження даних. Після обробки запиту сервер формує відповідь у форматі JSON та повертає її клієнтському застосунку. Використання такого механізму дозволяє оновлювати інформацію без повного перезавантаження сторінки та забезпечує коректну взаємодію між усіма модулями системи [32, 39].

Програмний інтерфейс використовується під час перегляду каталогу товарів, відкриття сторінок продукції, оформлення замовлень, реєстрації користувачів та надсилання індивідуальних заявок. Усі операції виконуються через окремі маршрути API, кожен з яких відповідає за певний функціональний модуль системи.

Основні маршрути REST API наведено в таблиці 2.2.

Таблиця 2.2

Основні маршрути REST API

Метод	Маршрут	Призначення
GET	/api/products	Отримання списку товарів
GET	/api/products/{id}	Отримання інформації про товар
POST	/api/orders	Створення замовлення
POST	/api/custom-orders	Створення індивідуального замовлення
POST	/api/register	Реєстрація користувача
POST	/api/login	Авторизація користувача

Наведений набір маршрутів охоплює основні сценарії роботи користувача із системою. Через API здійснюється отримання даних каталогу, перегляд інформації про товари, оформлення замовлень та взаємодія з обліковими записами користувачів.

Використання REST API спрощує взаємодію між окремими рівнями системи та дозволяє розвивати клієнтську і серверну частини незалежно одна від

одної. Такий підхід широко застосовується під час розробки сучасних вебдодатків і добре підходить для реалізації систем електронної комерції [38, 39].

2.4 Проєктування UML-діаграми варіантів використання

Для визначення функціональних можливостей системи та ролей користувачів було побудовано UML-діаграму варіантів використання. Така діаграма дозволяє відобразити взаємодію користувачів із вебдодатком та визначити основні сценарії роботи системи [11, 13].

У проєкті передбачено три ролі: гість, користувач та адміністратор. Кожна роль має власний набір функцій відповідно до рівня доступу та призначення в системі.

Гість може переглядати головну сторінку, каталог продукції, виконувати пошук товарів та відкривати сторінки окремих товарів. Для отримання додаткових можливостей передбачено функції реєстрації та авторизації. Такий підхід дозволяє ознайомитися з асортиментом продукції без створення облікового запису.

Після авторизації користувач отримує доступ до функцій оформлення замовлень. Він може додавати товари до кошика, змінювати їх кількість, видаляти окремі позиції та переходити до оформлення покупки. Також користувачу доступний особистий кабінет і можливість створення індивідуального замовлення на формування квіткової композиції.

Функція індивідуального замовлення використовується у випадках, коли необхідно сформувати букет за власними параметрами. Під час створення заявки користувач може вказати бажані квіти, орієнтовний бюджет, дату доставки та додаткові побажання щодо композиції.

Адміністратор відповідає за керування вмістом системи та обробку отриманих даних. Для нього передбачено можливість додавання, редагування та видалення товарів, перегляду замовлень, роботи з індивідуальними заявками та зміни статусів замовлень. Окремо реалізовано доступ до інформації, необхідної для контролю роботи вебсайту.

UML-діаграму варіантів використання наведено на рисунку 2.4.

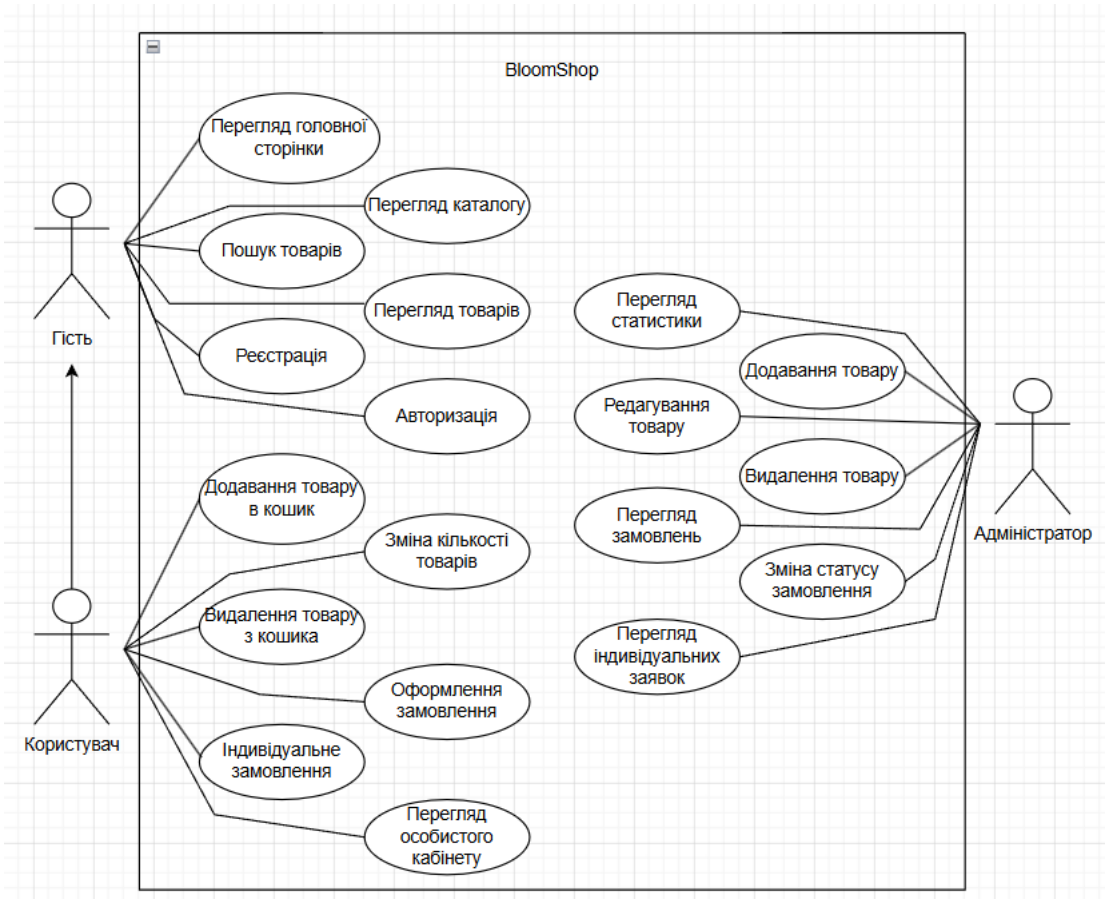


Рисунок 2.4. UML-діаграма варіантів використання інформаційної системи комерційної реалізації квітів

Висновки до розділу 2

У другому розділі виконано проєктування інформаційної системи комерційної реалізації квітів. Визначено структуру вебсайту, склад основних сторінок та логіку переходів між ними. Розроблено схему навігації, яка охоплює головну сторінку, каталог товарів, сторінку товару, кошик, оформлення замовлення, особистий кабінет користувача та адміністративну панель.

Під час проєктування користувацького інтерфейсу визначено структуру каталогу продукції, що включає п’ять категорій товарів: букети, квіти в коробках, монобукети, рослини та подарунки. Передбачено механізми пошуку, фільтрації товарів і переходу до сторінки детального перегляду продукції.

Для клієнтської частини спроєктовано frontend-архітектуру на основі React. Визначено компонентну структуру застосунку, організацію директорій проєкту та систему клієнтської маршрутизації з використанням React Router. Такий підхід забезпечує логічний поділ інтерфейсу на окремі функціональні модулі та спрощує підтримку програмного коду.

У межах проєктування серверної частини визначено архітектуру взаємодії між клієнтським і серверним рівнями системи. Для обміну даними передбачено використання REST API, а для реалізації серверної логіки — Flask. Також спроєктовано структуру бази даних SQLite, яка включає таблиці користувачів, товарів, замовлень, позицій замовлень, категорій та індивідуальних заявок.

Крім того, побудовано UML-діаграму варіантів використання, яка відображає функціональні можливості системи для гостя, зареєстрованого користувача та адміністратора. Результати проєктування стали основою для реалізації програмних модулів вебдодатку, що розглядаються в наступному розділі.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБСАЙТУ

Після завершення етапу проєктування було виконано реалізацію вебсайту комерційної реалізації квіткової продукції. На цьому етапі створено клієнтську та серверну частини системи, реалізовано основні функціональні модулі та забезпечено їх взаємодію між собою.

Клієнтська частина вебдодатку розроблена з використанням бібліотеки React та засобів стилізації Tailwind CSS, що дозволило реалізувати адаптивний користувацький інтерфейс і компонентну структуру застосунку [18, 26, 28]. Для організації навігації між сторінками використано React Router DOM [27].

Серверна частина реалізована на основі мікрофреймворку Flask. Для зберігання інформації про товари, користувачів, замовлення та індивідуальні заявки використовується база даних SQLite, взаємодія з якою здійснюється через SQLAlchemy ORM [31–34].

У межах розділу розглянуто реалізацію основних сторінок вебсайту, механізми роботи каталогу продукції, сторінки товару, кошика, оформлення замовлення, авторизації користувачів та адміністративної панелі. Також наведено результати тестування реалізованого функціоналу та оцінено коректність роботи окремих модулів системи.

3.1 Реалізація клієнтської частини вебсайту

Клієнтська частина забезпечує взаємодію користувача із системою та доступ до основних функцій вебдодатку. Під час реалізації особлива увага приділялася зручності навігації, адаптивності інтерфейсу та швидкому доступу до функціональних можливостей системи.

Головна сторінка виконує роль центрального елемента інтерфейсу та забезпечує доступ до основних розділів вебсайту. На ній розміщено навігаційне меню, рекламний банер, блок категорій продукції, популярні товари, інформацію

про переваги сервісу, форму індивідуального замовлення та відгуки користувачів.

Зовнішній вигляд головної сторінки вебсайту наведено на рисунку 3.1.

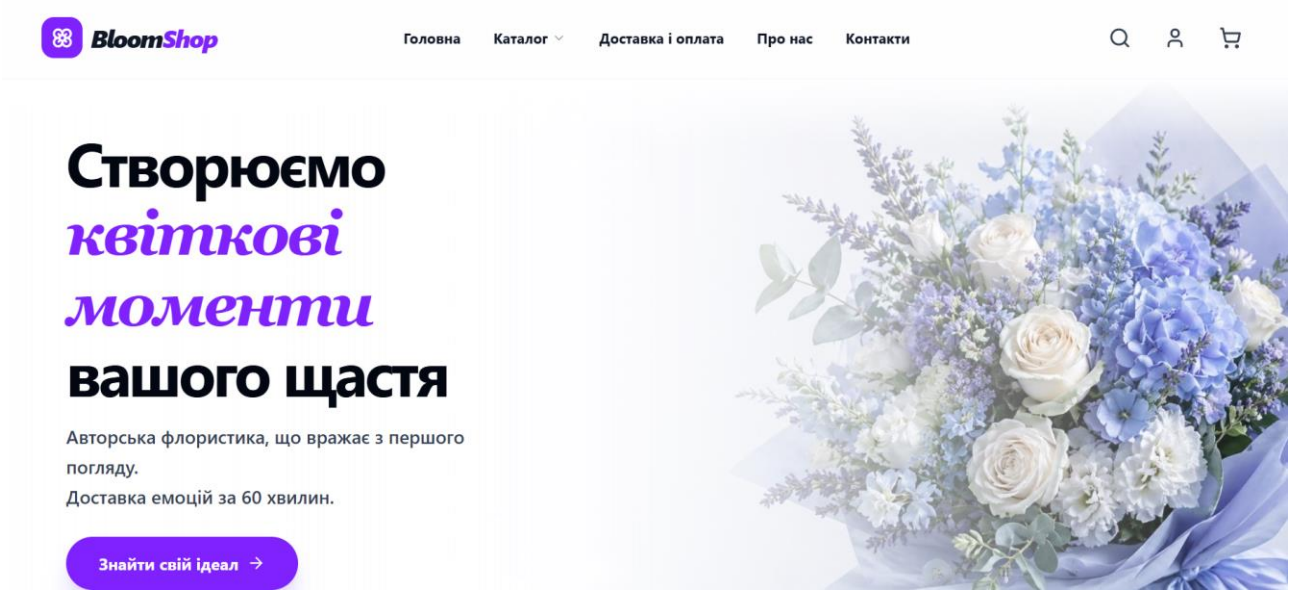


Рисунок 3.1. Головна сторінка вебсайту BloomShop

Як показано на рисунку 3.1, у верхній частині сторінки розміщено навігаційне меню з переходами до каталогу продукції, інформаційних розділів та функцій користувача. Праворуч знаходяться елементи пошуку, авторизації та кошика, що забезпечують швидкий доступ до основних можливостей системи.

Центральну частину сторінки займає рекламний банер із короткою інформацією про сервіс та кнопкою переходу до каталогу товарів. Нижче розташовано блок категорій продукції, який дозволяє перейти до відповідних груп товарів без додаткового пошуку.

Для демонстрації асортименту реалізовано секцію популярних товарів, що формується на основі даних каталогу та відображає найбільш актуальні позиції. Такий підхід спрощує ознайомлення користувача з асортиментом і відповідає рекомендаціям щодо побудови інтерфейсів електронної комерції [3, 4].

Крім цього, на головній сторінці реалізовано блок переваг сервісу, форму індивідуального замовлення букета та секцію відгуків. Використання зазначених

елементів дозволяє надати користувачу додаткову інформацію про можливості вебсайту та спростити взаємодію із сервісом.

Таким чином, головна сторінка забезпечує доступ до основних функцій системи та виконує роль початкової точки взаємодії користувача з вебдодатком.

3.1.1 Реалізація каталогу товарів

Каталог товарів є одним із основних функціональних модулів розробленого вебдодатку, оскільки саме через нього користувач отримує доступ до асортименту продукції. Реалізацію каталогу виконано за допомогою React із використанням компонентного підходу та клієнтської обробки даних, що дозволяє оновлювати вміст сторінки без її повного перезавантаження [18, 26].

Зовнішній вигляд каталогу товарів наведено на рисунку 3.2.

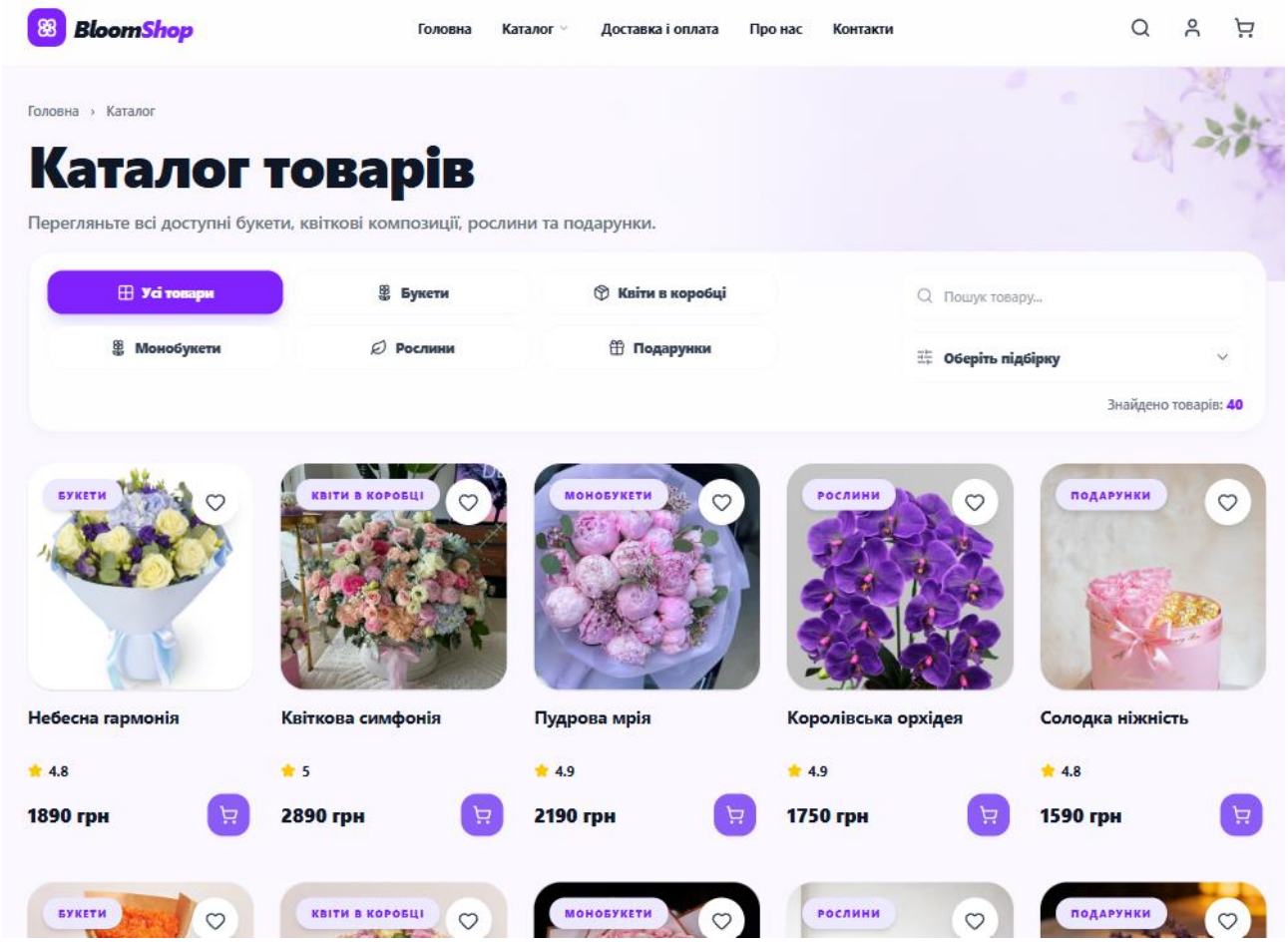


Рисунок 3.2. Каталог товарів вебсайту

Ми бачимо що на рисунку 3.2, у верхній частині сторінки розміщено навігаційний ланцюжок, назву розділу та інформацію про призначення каталогу. Нижче знаходиться панель керування, яка містить категорії товарів, поле пошуку та засоби сортування продукції.

Для впорядкування асортименту реалізовано поділ товарів на п’ять категорій: букети, квіти в коробках, монобукети, рослини та подарунки. Перемикання між категоріями здійснюється без перезавантаження сторінки, що забезпечується засобами React та механізмами керування станом компонентів [18].

У каталозі реалізовано пошук товарів за назвою. Під час введення тексту список продукції автоматично оновлюється відповідно до введеного запиту. Крім того, користувач може використовувати сортування для швидкого відбору потрібних позицій каталогу.

Для демонстрації роботи системи сформовано асортимент із сорока товарів, які рівномірно розподілені між усіма категоріями. Інформація про товари містить назву, ціну, рейтинг, опис та набір фотографій. Дані відображаються у вигляді карток, що забезпечують компактне представлення продукції та спрощують перегляд каталогу.

Кожна картка товару містить основне зображення, назву продукції, рейтинг, вартість та елементи взаємодії з користувачем. Після вибору товару здійснюється перехід на сторінку детального перегляду, де відображається розширена інформація про обрану позицію.

Під час реалізації каталогу також враховано вимоги адаптивності. Розташування карток автоматично змінюється залежно від розміру екрана, що забезпечує коректне відображення сторінки на персональних комп’ютерах, планшетах і мобільних пристроях [20, 28].

Реалізований каталог забезпечує зручний перегляд асортименту, підтримує пошук, сортування та фільтрацію товарів і є основним інструментом взаємодії користувача з продукцією вебдодатку.

3.1.2 Реалізація сторінки товару

Після вибору товару в каталозі користувач переходить на сторінку детального перегляду продукції. Ця сторінка призначена для відображення повної інформації про товар та забезпечує можливість його додавання до кошика.

Зовнішній вигляд сторінки товару наведено на рисунку 3.3.

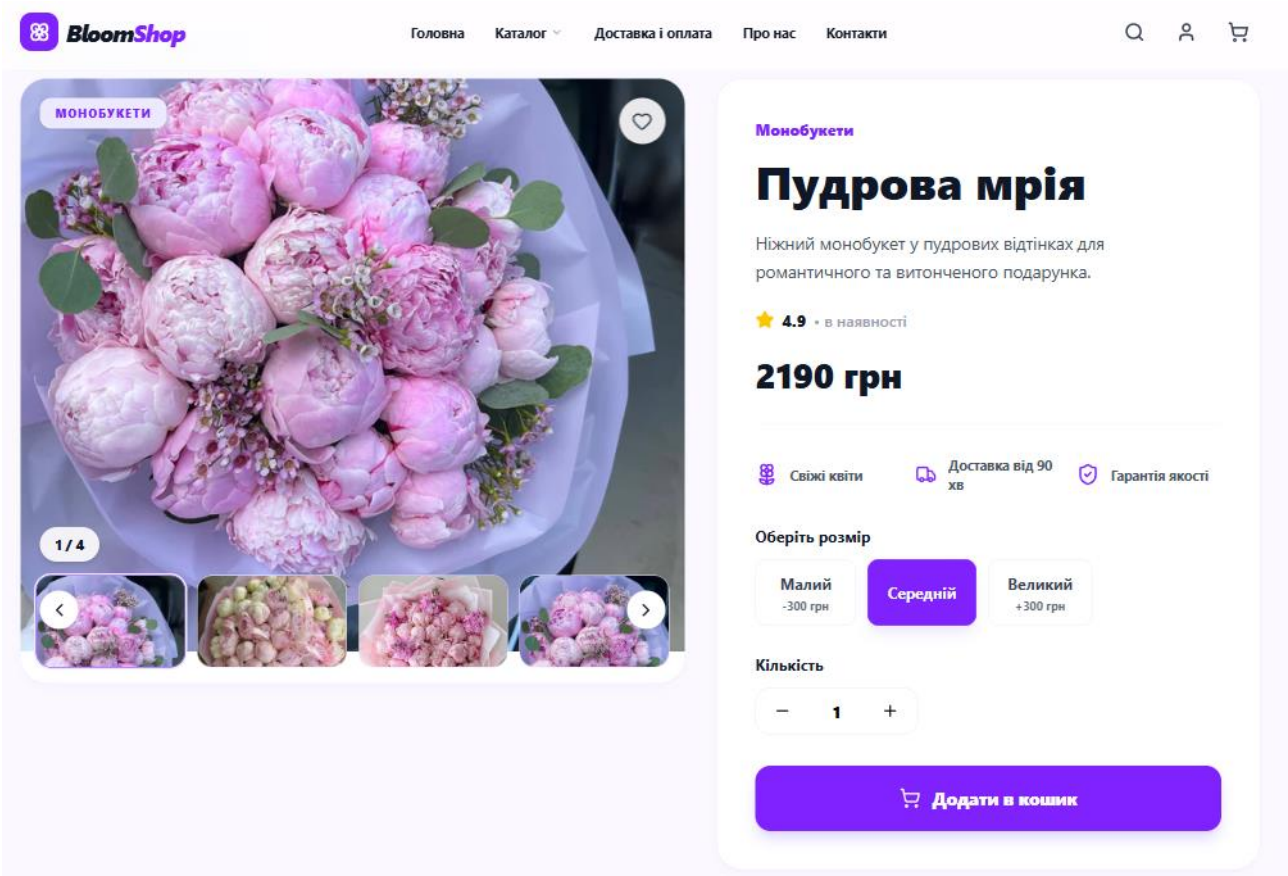


Рисунок 3.3. Сторінка окремого товару

На рисунку 3.3 показано основну частину сторінки займає галерея зображень товару. Для кожної позиції може використовуватися декілька фотографій, між якими користувач може перемикатися за допомогою мініатюр або кнопок навігації. Поточне зображення відображається у збільшеному вигляді, що дозволяє детальніше ознайомитися із зовнішнім виглядом композиції.

У правій частині сторінки розміщено основну інформацію про товар: назву, короткий опис, рейтинг та вартість. Додатково відображаються характеристики, пов’язані з якістю продукції та умовами доставки.

Для окремих товарів реалізовано вибір розміру композиції. Користувач може обрати малий, середній або великий варіант букета. Після зміни розміру виконується автоматичний перерахунок вартості продукції, що дозволяє відразу бачити актуальну ціну замовлення.

На сторінці також передбачено механізм вибору кількості товару. Збільшення або зменшення кількості здійснюється за допомогою окремих елементів керування без необхідності ручного введення значень.

Для додавання товару до списку вподобаних реалізовано функцію «Обране», яка дозволяє зберігати вибрані позиції для подальшого перегляду. Після завершення вибору параметрів користувач може додати товар до кошика та перейти до оформлення замовлення.

Реалізація сторінки товару виконана з використанням компонентів React та механізмів динамічного оновлення даних [18, 26]. Це забезпечує швидку зміну вмісту сторінки та взаємодію користувача з елементами інтерфейсу без перезавантаження браузера.

Таким чином, сторінка товару забезпечує повний набір функцій для ознайомлення з продукцією, вибору параметрів замовлення та подальшого додавання товару до кошика.

3.1.3 Реалізація кошика та оформлення замовлення

Для роботи із замовленнями в системі реалізовано окремі сторінки кошика та оформлення покупки. Їх призначення полягає у формуванні списку вибраних товарів, розрахунку вартості замовлення та введенні даних, необхідних для доставки продукції.

Після додавання товарів до кошика інформація зберігається у глобальному стані застосунку та доступна на всіх сторінках вебсайту. Користувач може

переглядати перелік вибраної продукції, змінювати кількість одиниць товару, видаляти окремі позиції та контролювати загальну суму замовлення.

Зовнішній вигляд сторінки кошика наведено на рисунку 3.4.

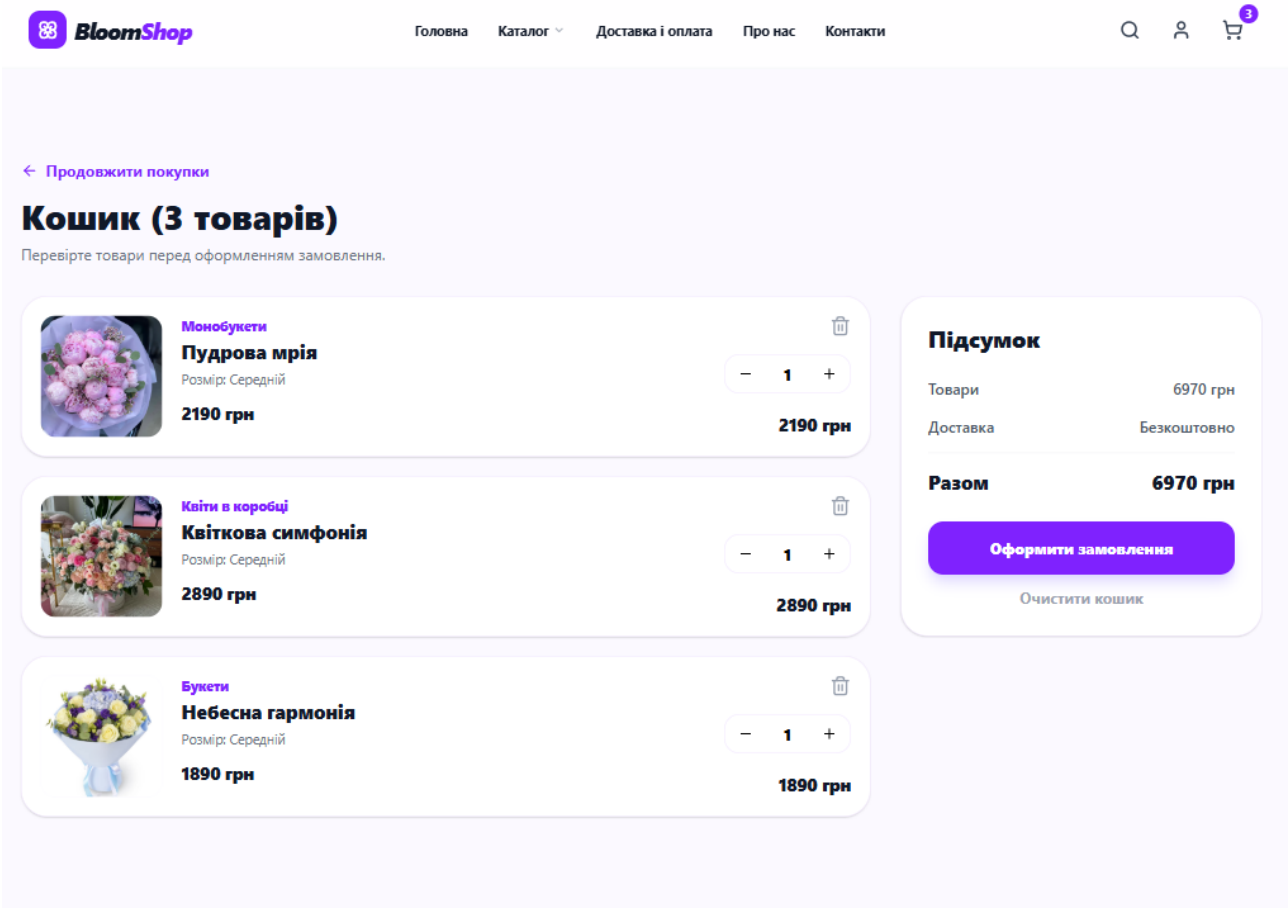
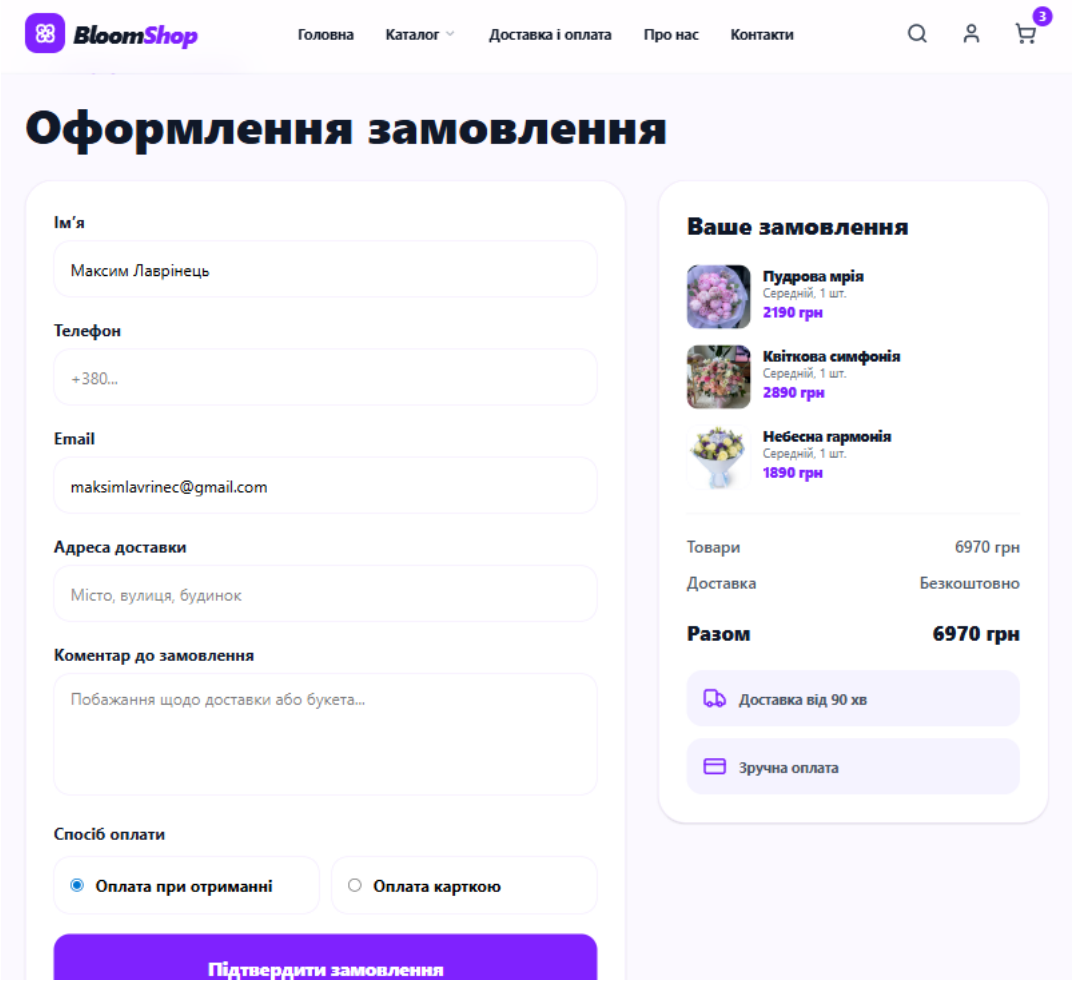


Рисунок 3.4. Сторінка кошика

На сторінці відображається список доданих товарів із фотографіями, назвами, обраними параметрами та вартістю. Для кожної позиції реалізовано зміну кількості товару за допомогою окремих елементів керування, а також можливість видалення товару з кошика. У правій частині сторінки розташовано блок підсумку замовлення, де автоматично розраховується загальна вартість покупки та відображається інформація про доставку.

Після перевірки складу замовлення користувач переходить до сторінки оформлення покупки. Передача даних між сторінками здійснюється через механізми керування станом React та маршрутизацію вебдодатку [18, 27].

Зовнішній вигляд сторінки оформлення замовлення наведено на рисунку 3.5.



3.5 - Сторінка оформлення замовлення.

На сторінці оформлення замовлення розміщено форму введення контактних даних користувача. Для створення заявки необхідно вказати ім'я, номер телефону, адресу електронної пошти, адресу доставки та додатковий коментар. Також реалізовано вибір способу оплати та відображення короткого підсумку замовлення із переліком товарів і загальною сумою покупки.

Після підтвердження замовлення система формує заявку та передає її для подальшої обробки серверною частиною. Реалізований механізм дозволяє виконати повний цикл оформлення покупки — від вибору товару до надсилання замовлення без переходу на сторонні ресурси.

Таким чином, модулі кошика та оформлення замовлення забезпечують роботу з вибраними товарами, автоматичний розрахунок вартості покупки та введення інформації, необхідної для подальшого опрацювання замовлення.

3.1.4 Реалізація конструктора та індивідуального замовлення букета

Для розширення функціональних можливостей вебдодатку реалізовано модуль індивідуального замовлення букета. Його призначення полягає у формуванні заявки на створення композиції відповідно до побажань користувача без вибору готового товару з каталогу.

Зовнішній вигляд модуля наведено на рисунку 3.6.

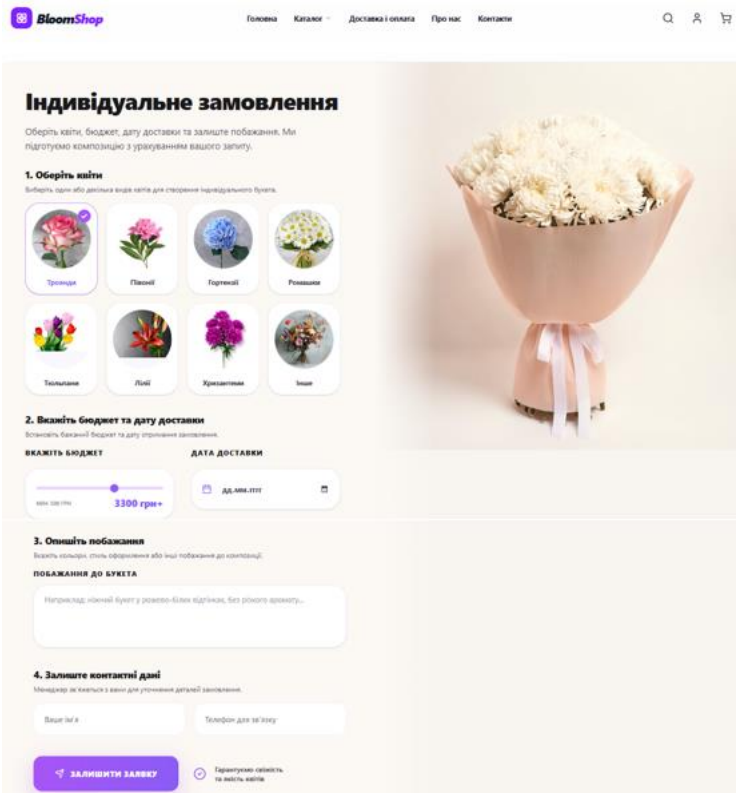


Рисунок 3.6. Модуль індивідуального замовлення букета

На першому етапі користувач обирає види квітів, які бажає включити до майбутньої композиції. Для цього реалізовано набір карток із фотографіями та назвами квітів. Підтримується вибір декількох варіантів одночасно, що дозволяє сформувати індивідуальне замовлення відповідно до власних уподобань.

Наступним етапом є визначення бюджету та дати доставки. Для вибору бюджету використовується інтерактивний повзунок, який дозволяє встановити бажану вартість майбутнього букета. Дата доставки задається через окремий елемент вибору дати.

Додатково реалізовано текстове поле для введення побажань до композиції. У ньому користувач може вказати особливості оформлення букета, кольорову гаму, побажання щодо складу або іншу інформацію, необхідну для виконання замовлення.

Для завершення формування заявки передбачено введення контактних даних, необхідних для зворотного зв’язку. Після заповнення всіх полів інформація передається до серверної частини та зберігається як індивідуальне замовлення користувача.

Реалізація конструктора дозволяє врахувати специфіку продажу квіткової продукції, де значна частина клієнтів віддає перевагу композиціям, сформованим за власними побажаннями. Наявність такого модуля розширює можливості вебдодатку та забезпечує додатковий механізм взаємодії між клієнтом і магазином.

3.2 Реалізація додаткового функціоналу системи

Окрім основних модулів каталогу, кошика та оформлення замовлень, у вебдодатку реалізовано додатковий функціонал, спрямований на підвищення зручності роботи користувачів. До нього належать модуль авторизації, реєстрації та особистий кабінет користувача. Використання таких компонентів дозволяє зберігати персональні дані, забезпечувати швидший доступ до оформлення замовлень та створювати більш зручний механізм взаємодії із системою [18, 26].

3.2.1 Реалізація авторизації та особистого кабінету користувача

Для роботи з обліковими записами реалізовано модуль авторизації та реєстрації користувачів. Його основою є модальне вікно, яке відкривається без переходу на окрему сторінку та містить форми входу і створення нового облікового запису. Такий підхід відповідає сучасним рекомендаціям щодо проєктування користувацьких інтерфейсів та дозволяє скоротити кількість дій, необхідних для авторизації користувача [3, 4].

Зовнішній вигляд модуля авторизації та особистого кабінету наведено на рисунках 3.7 – 3.8..

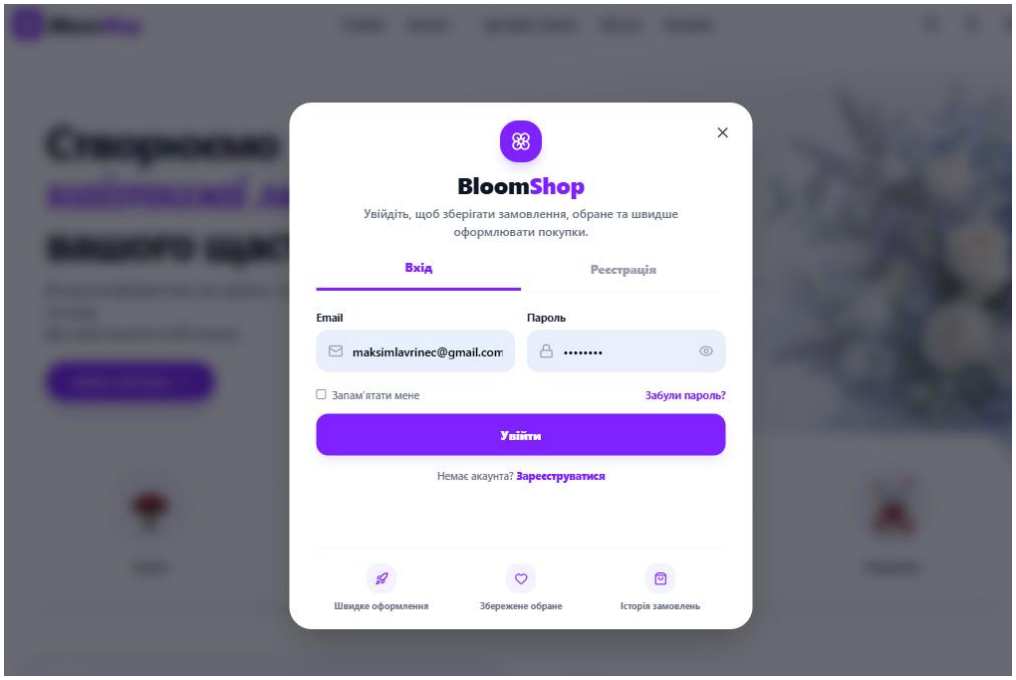


Рисунок 3.7. Модуль авторизації

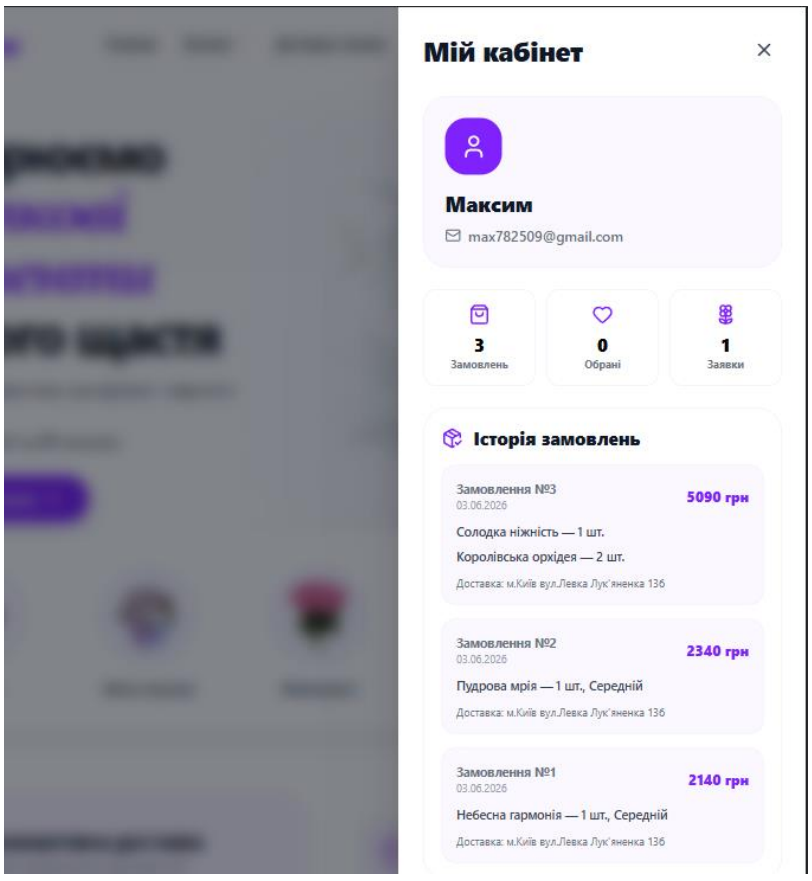


Рисунок 3.8. Особистий кабінет користувача

Модуль авторизації містить поля для введення електронної пошти та пароля, а також можливість переходу між режимами входу та реєстрації. Після успішної авторизації користувач отримує доступ до персональних функцій системи.

Особистий кабінет реалізовано у вигляді бічної панелі, яка відкривається поверх основного інтерфейсу вебсайту. У кабінеті відображаються основні відомості про користувача, кількість оформлених замовлень, список обраних товарів та статистика індивідуальних заявок. Також передбачено блок історії замовлень, де відображаються раніше оформлені покупки із зазначенням складу замовлення, адреси доставки та його вартості.

Для збереження даних авторизації використовується механізм Local Storage, що дозволяє зберігати інформацію між сеансами роботи браузера. Реалізація такого підходу забезпечує підтримку авторизації без постійного повторного введення облікових даних та створює основу для подальшої інтеграції із серверною частиною системи [21, 26].

Реалізований модуль дозволяє організувати персоналізовану взаємодію користувача з вебдодатком та забезпечує зручний доступ до інформації, пов’язаної із замовленнями та профілем користувача.системи.

3.2.2 Реалізація адміністративної панелі

Для керування вмістом системи реалізовано адміністративну панель, яка забезпечує доступ до основних операцій з товарами, замовленнями та індивідуальними заявками користувачів. Наявність окремого адміністративного модуля дозволяє розмежувати функції звичайних користувачів та адміністратора, що відповідає принципам побудови інформаційних систем із різними рівнями доступу [5, 9].

Зовнішній вигляд адміністративної панелі наведено на рисунку 3.9.

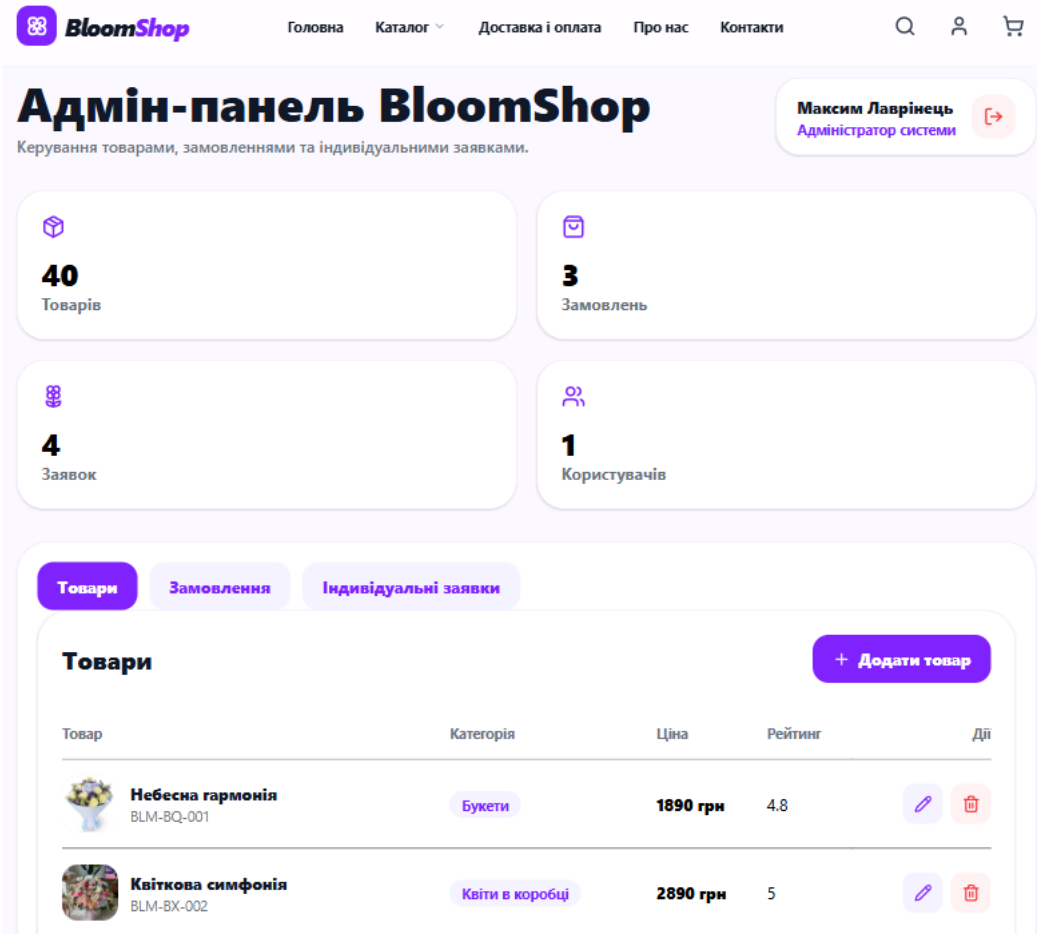


Рисунок 3.9. Адміністративна панель вебсайту

Адміністративний інтерфейс містить інформаційні блоки зі статистичними показниками системи, зокрема кількістю товарів, оформлених замовлень, індивідуальних заявок та зареєстрованих користувачів. Це дозволяє отримувати загальну інформацію про стан системи без переходу між окремими сторінками.

Основна частина панелі реалізована у вигляді вкладок, які забезпечують доступ до керування товарами, перегляду замовлень та роботи з індивідуальними заявками. Для кожного товару відображаються основні відомості: назва, категорія, вартість і рейтинг. Передбачено окремі інструменти для додавання нових позицій, редагування наявної інформації та видалення товарів із каталогу.

Модуль роботи із замовленнями дозволяє переглядати отримані заявки та контролювати їх обробку. Окремо реалізовано перегляд індивідуальних

замовлень, сформованих через конструктор букетів. Це спрощує опрацювання запитів клієнтів та забезпечує централізоване керування всіма даними системи.

Під час реалізації адміністративної частини використовувався той самий компонентний підхід, що й для інших сторінок вебдодатку. Завдяки цьому вдалося забезпечити єдиний стиль оформлення інтерфейсу та спростити подальший розвиток функціоналу системи [18, 26].

Реалізована адміністративна панель забезпечує виконання основних операцій з керування даними та створює основу для подальшого розширення функціональних можливостей серверної частини вебдодатку.

3.3 Тестування вебсайту

Після завершення реалізації основних модулів було проведено тестування вебсайту з метою перевірки коректності роботи функціональних компонентів, навігації між сторінками та взаємодії користувача із системою. Тестування виконувалося відповідно до функціональних вимог, сформованих на етапі аналізу та проєктування системи [5, 6].

У процесі перевірки особлива увага приділялася основним сценаріям використання вебдодатку. Було протестовано роботу каталогу товарів, пошуку та фільтрації продукції, перегляд сторінок окремих товарів, додавання товарів до кошика, зміну кількості позицій у замовленні та оформлення покупки. Також перевірялася коректність роботи модулів авторизації, особистого кабінету користувача, індивідуального замовлення букета та адміністративної панелі.

Окремо виконано перевірку клієнтської маршрутизації, реалізованої за допомогою React Router. Під час тестування підтверджено коректний перехід між основними сторінками системи без повного перезавантаження браузера, що відповідає принципам побудови SPA-додатків [18, 26, 27].

Крім функціонального тестування було проведено перевірку адаптивності інтерфейсу. Робота вебсайту перевірялася на різних розмірах екранів із метою підтвердження коректного відображення сторінок, навігаційних елементів, карток товарів та форм введення даних. Результати перевірки показали коректну

роботу інтерфейсу як на персональних комп’ютерах, так і на мобільних пристроях [3, 4].

3.3.1 Перевірка працездатності основних функцій

Для оцінки працездатності розробленого вебдодатку було проведено функціональне тестування основних модулів системи. Перевірка виконувалася шляхом відтворення типових сценаріїв взаємодії користувача з вебсайтом, зокрема перегляду каталогу, пошуку товарів, формування замовлення та роботи додаткових функціональних модулів [5, 6].

Таблиця 3.1

Результати тестування функціональних модулів

Функціональний модуль	Результат перевірки
Перегляд головної сторінки	Виконується коректно
Перегляд каталогу товарів	Виконується коректно
Пошук товарів	Виконується коректно
Фільтрація за категоріями	Виконується коректно
Перегляд сторінки товару	Виконується коректно
Додавання товару до кошика	Виконується коректно
Оформлення замовлення	Виконується коректно
Індивідуальне замовлення букета	Виконується коректно
Авторизація користувача	Виконується коректно
Робота особистого кабінету	Виконується коректно

Під час тестування помилок, які б перешкоджали виконанню основних сценаріїв роботи користувача, виявлено не було. Перевірка підтвердила коректність роботи навігації між сторінками, функцій пошуку та фільтрації товарів, механізму формування кошика, оформлення замовлень, а також модулів авторизації та особистого кабінету.

Отримані результати свідчать про відповідність реалізованого функціоналу вимогам, сформованим під час аналізу та проєктування системи, а також підтверджують працездатність основних компонентів вебдодатку [5, 6, 18].

3.3.2 Аналіз результатів тестування

За результатами проведеного тестування встановлено, що реалізовані функціональні модулі працюють стабільно та забезпечують виконання основних сценаріїв взаємодії користувача із системою. У процесі перевірки підтверджено коректну роботу навігації між сторінками, маршрутизації вебдодатку та передачі даних між окремими компонентами інтерфейсу [5, 6].

Перевірка каталогу товарів показала правильність роботи механізмів пошуку, фільтрації та відображення продукції за категоріями. Під час тестування сторінок товарів підтверджено коректне завантаження інформації про продукцію, роботу галереї зображень, вибір розміру композиції та додавання товарів до кошика.

Тестування кошика та оформлення замовлення засвідчило правильність розрахунку вартості замовлення, оновлення кількості товарів і передачі введених користувачем даних до відповідних модулів системи. Також успішно пройшли перевірку форми індивідуального замовлення букета, механізми авторизації та особистий кабінет користувача.

Окрему увагу було приділено перевірці адаптивності інтерфейсу. Під час тестування на різних роздільних здатностях екрана встановлено, що сторінки каталогу, товарів, кошика та інших модулів коректно відображаються на персональних комп’ютерах, планшетах і мобільних пристроях. Використання React та Tailwind CSS забезпечило стабільну роботу інтерфейсу та коректне масштабування елементів сторінки [18, 28].

Отримані результати підтверджують відповідність реалізованого вебдодатку функціональним вимогам, сформованим у першому розділі роботи. Проведене тестування дозволило переконатися у працездатності основних модулів системи та готовності програмного продукту до подальшого використання й розвитку [5, 6].

Висновки до розділу 3

У третьому розділі виконано реалізацію та тестування вебсайту комерційної реалізації квітів відповідно до архітектурних рішень, розроблених у другому розділі роботи.

У ході реалізації створено клієнтську частину вебдодатку на основі React та Tailwind CSS. Розроблено головну сторінку, каталог продукції, сторінки товарів, кошик, форму оформлення замовлення, модуль індивідуального замовлення букета, систему авторизації користувачів, особистий кабінет та адміністративну панель. Також реалізовано механізми пошуку, фільтрації товарів за категоріями, перегляду детальної інформації про продукцію та формування замовлення.

Окрему увагу приділено реалізації функціоналу індивідуального замовлення букета, який дозволяє користувачу обирати тип квітів, бюджет, дату доставки та залишати власні побажання щодо майбутньої композиції. Даний модуль враховує специфіку предметної області та розширює можливості взаємодії користувача із вебсайтом.

Під час тестування перевірено роботу основних функціональних модулів системи, включаючи навігацію між сторінками, каталог товарів, пошук і фільтрацію продукції, кошик, оформлення замовлень, авторизацію користувачів та особистий кабінет. Результати перевірки підтвердили коректність роботи реалізованого функціоналу та відсутність критичних помилок під час виконання основних сценаріїв використання.

Проведене тестування також підтвердило коректне відображення інтерфейсу на різних типах пристроїв, що забезпечує зручну роботу із вебдодатком на персональних комп’ютерах, планшетах і смартфонах.

Отримані результати свідчать про відповідність реалізованого вебдодатку функціональним вимогам, визначеним на етапі аналізу та проєктування системи.

ВИСНОВКИ

У цій кваліфікаційній роботі виконано розробку інтерактивного вебсайту комерційної реалізації квітів, призначеного для перегляду асортименту продукції, пошуку товарів, оформлення замовлень та взаємодії користувача з основними функціональними модулями системи.

Під час виконання роботи проведено аналіз предметної області електронної комерції у сфері продажу квіткової продукції та досліджено особливості функціонування сучасних інтернет-магазинів. Виконано аналіз існуючих галузевих рішень, визначено їх переваги та недоліки, а також сформовано функціональні та нефункціональні вимоги до розроблюваної системи.

У процесі проєктування розроблено структуру вебсайту, визначено основні сценарії взаємодії користувачів із системою, спроєктовано навігацію між сторінками, структуру каталогу товарів, архітектуру клієнтської та серверної частин, модель бази даних і програмний інтерфейс взаємодії між компонентами системи. Для формалізації вимог та функціональних можливостей було побудовано UML-діаграму варіантів використання.

Для реалізації клієнтської частини використано бібліотеку React та фреймворк Tailwind CSS. Застосування компонентного підходу дозволило побудувати інтерфейс у вигляді незалежних функціональних модулів, що спрощує підтримку та подальший розвиток проєкту. Для маршрутизації сторінок використано React Router, що забезпечило реалізацію односторінкового вебдодатку із швидким переходом між розділами системи.

Під час реалізації створено головну сторінку, каталог продукції, сторінки окремих товарів, кошик, форму оформлення замовлення, систему пошуку та фільтрації товарів, модуль індивідуального замовлення букета, систему авторизації та особистий кабінет користувача. Також реалізовано адміністративну панель, яка забезпечує керування товарами, замовленнями та індивідуальними заявками.

Для серверної частини обрано мікрофреймворк Flask, а для зберігання даних — систему керування базами даних SQLite. Взаємодія між клієнтською та серверною частинами організована за допомогою REST API, що забезпечує обмін даними між компонентами системи та підтримує клієнт-серверну архітектуру вебдодатку.

У процесі тестування перевірено працездатність основних функціональних модулів системи. Підтверджено коректну роботу каталогу товарів, пошуку, фільтрації, сторінок товарів, кошика, оформлення замовлень, авторизації користувачів, особистого кабінету та адміністративної панелі. Також перевірено адаптивність інтерфейсу на різних типах пристроїв, що підтвердило коректне відображення сторінок на персональних комп’ютерах, планшетах і смартфонах.

Таким чином, поставлену мету роботи досягнуто. У результаті виконання кваліфікаційної роботи створено інтерактивний вебсайт комерційної реалізації квітів, який реалізує основні функції інтернет-магазину та відповідає визначеним функціональним і нефункціональним вимогам. Отримані результати можуть бути використані для подальшого розширення функціональних можливостей системи, інтеграції платіжних сервісів, автоматизації обробки замовлень та розвитку серверної частини вебдодатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Laudon K. C., Traver C. G. E-commerce 2023–2024: Business, Technology, Society. 18th ed. Harlow : Pearson, 2023.

2. Turban E., Pollard C., Wood G. Information Technology for Management: Driving Digital Transformation to Increase Local and Global Performance, Growth and Sustainability. 12th ed. Hoboken : Wiley, 2021. 656 p.

3. Nielsen J. Usability Engineering. Amsterdam ; Boston : Morgan Kaufmann, 1994. 362 p.

4. Krug S. Don’t Make Me Think, Revisited: A Common Sense Approach to Web Usability. 3rd ed. Berkeley : New Riders, 2014. 212 p.

5. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner’s Approach. 9th ed. New York : McGraw-Hill Education, 2020.

6. Sommerville I. Software Engineering. 10th ed. Harlow : Pearson, 2016.

7. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2003. 533 p.

8. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Reading : Addison-Wesley, 1994. 395 p.

9. Martin R. C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.

10. Freeman E., Robson E. Head First Design Patterns. 2nd ed. Sebastopol : O’Reilly Media, 2020. 694 p.

11. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed. Boston : Addison-Wesley, 2004. 208 p.

12. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. Sebastopol : O’Reilly Media, 2020. 432 p.

13. Шаховська Н. Б., Литвин В. В. Проектування інформаційних систем : навч. посіб. Львів : Магнолія 2006, 2021. 380 с.

14. Пасічник В. В., Резніченко В. А. Організація баз даних та знань : підручник. Київ : Видавнича група BHV, 2006. 384 с.

15. Завадський І. О. Основи баз даних : навч. посіб. Київ : Видавець І. О. Завадський, 2011. 192 с.

16. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2nd ed. Sebastopol : O'Reilly Media, 2020. 310 p.

17. Машкіна І. В., Абрамов В. О., Носенко Т. І., Іваніченко Є. В., Яскевич В. О. Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп'ютерні науки для здобувачів вищої освіти денної форми навчання. Київ : Київський столичний університет імені Бориса Грінченка, 2024.

18. Яскевич В. О. JavaScript-бібліотека React : навч. посіб. для студентів спеціальності «Комп'ютерні науки». Київ : Київський університет імені Бориса Грінченка, 2023.

19. Mozilla Developer Network. HTML: HyperText Markup Language. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 20.02.2026).

20. Mozilla Developer Network. CSS: Cascading Style Sheets. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 22.02.2026).

21. Mozilla Developer Network. JavaScript Guide. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 25.02.2026).

22. WHATWG. HTML Living Standard. URL: https://html.spec.whatwg.org (дата звернення: 01.03.2026).

23. ECMA International. ECMAScript Language Specification. URL: https://tc39.es/ecma262 (дата звернення: 03.03.2026).

24. Duckett J. HTML and CSS: Design and Build Websites. Indianapolis : Wiley, 2011.

25. Duckett J. JavaScript and jQuery: Interactive Front-End Web Development. Indianapolis : Wiley, 2014. 640 p.

26. React Documentation. URL: https://react.dev (дата звернення: 10.03.2026).

27. React Router Documentation. URL: https://reactrouter.com (дата звернення: 12.03.2026).

28. Tailwind CSS Documentation. URL: https://tailwindcss.com/docs (дата звернення: 15.03.2026).

29. Vite Documentation. URL: https://vite.dev (дата звернення: 18.03.2026).

30. Wieruch R. The Road to React. URL: https://www.roadtoreact.com/ (дата звернення: 20.03.2026).

31. Flask-SQLAlchemy Documentation. URL: https://flask-sqlalchemy.readthedocs.io/en/stable/ (дата звернення: 23.03.2026).

32. Flask Documentation. URL: https://flask.palletsprojects.com (дата звернення: 25.03.2026).

33. SQLAlchemy Documentation. URL: https://docs.sqlalchemy.org (дата звернення: 27.03.2026).

34. SQLite Documentation. URL: https://www.sqlite.org/docs.html (дата звернення: 30.03.2026).

35. Elmasri R., Navathe S. B. Fundamentals of Database Systems. 7th ed. Boston : Pearson, 2016.

36. Coronel C., Morris S. Database Systems: Design, Implementation, and Management. 13th ed. Boston : Cengage Learning, 2018. 816 p.

37. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York : McGraw-Hill Education, 2019.

38. Fielding R. T. Architectural Styles and the Design of Network-Based Software Architectures : dissertation. Irvine : University of California, 2000.

39. Richardson L., Amundsen M., Ruby S. RESTful Web APIs: Services for a Changing World. Sebastopol : O'Reilly Media, 2013. 406 p.

40. Statista. E-commerce Worldwide Statistics. URL: https://www.statista.com/topics/871/online-shopping/ (дата звернення: 10.04.2026).

41. Shopify. E-commerce Trends and Insights. URL: https://www.shopify.com/blog/ecommerce-trends (дата звернення: 12.04.2026).

42. Florium. URL: https://florium.ua (дата звернення: 05.04.2026).

43. Dicentra. URL: https://dicentra.ua (дата звернення: 10.04.2026).

44. DonPion. URL: https://donpion.ua (дата звернення: 15.04.2026).

45. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., ... & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science.

46. Машкіна, І. В., Абрамов, В. О., Носенко, Т. І., Іваніченко, Є. В., & Яскевич, В. О. (2024). Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп'ютерні науки для здобувачів вищої освіти денної форми навчання.

ДОДАТОК А

1. Маршрутизація вебсайту (App.jsx)

Фрагмент коду визначає основні маршрути клієнтської частини вебсайту BloomShop. Через ці маршрути користувач переходить між головною сторінкою, каталогом, сторінкою товару, кошиком, оформленням замовлення, особистим кабінетом та адміністративною панеллю.

```
<Routes>
  <Route path="/" element={<HomePage />} />
  <Route path="/catalog" element={<Catalog />} />
  <Route path="/product/:id" element={<ProductPage />} />
  <Route path="/cart" element={<Cart />} />
  <Route path="/checkout" element={<Checkout />} />
  <Route path="/profile" element={<Profile />} />
  <Route path="/privacy" element={<Privacy />} />
  <Route path="/admin" element={<Admin />} />
</Routes>
```

2. Логіка роботи кошика (CartContext.jsx)

Фрагмент коду демонструє додавання товару до кошика. Для кожного товару зберігаються назва, категорія, зображення, ціна, вибраний розмір та кількість. Якщо такий товар уже є в кошику, система збільшує його кількість, а не створює дубль.

```
const addToCart = (
  product,
  quantity = 1,
  selectedSize = null,
  finalPrice = null
) => {
  const cartProduct = {
    id: product.id,
    title: product.title,
    category: product.category,
    categoryLabel: product.categoryLabel,
    image: product.image,
    price: finalPrice || product.price,
    selectedSize,
    quantity,
  };

  setCartItems((prevItems) => {
    const existingItem = prevItems.find(
```

```
        (item) =>
          item.id === cartProduct.id &&
          item.selectedSize === cartProduct.selectedSize
      );

      if (existingItem) {
        return prevItems.map((item) =>
          item.id === cartProduct.id &&
          item.selectedSize === cartProduct.selectedSize
            ? { ...item, quantity: item.quantity + quantity }
            : item
        );
      }

      return [...prevItems, cartProduct];
    });
  };
```

3. Авторизація користувача (AuthContext.jsx)

Фрагмент коду показує логіку входу користувача в систему. Після успішної авторизації дані користувача зберігаються у стані застосунку та використовуються іншими компонентами вебсайту.

```
const login = async ({ email, password }) => {
  try {
    const data = await loginUser({ email, password });

    setUser(data.user);

    return {
      success: true,
      message: data.message || "Вхід виконано успішно",
    };
  } catch (error) {
    return {
      success: false,
      message: error.message || "Невірний email або пароль",
    };
  }
};
```

4. Створення індивідуального замовлення (CustomOrder.jsx)

Фрагмент коду відповідає за формування заявки на індивідуальний букет. У заявку передаються ім’я користувача, телефон, email, вибрані квіти, бюджет, дата доставки та побажання до композиції.

```
const customOrder = {
  name,
  phone,
  email: user?.email || "",
  selectedFlowers,
  budget,
  deliveryDate,
  wishes,
};

await createCustomOrder(customOrder);

setIsSent(true);
setName("");
setPhone("");
setWishes("");
setDeliveryDate("");
setBudget(3300);
setSelectedFlowers(["Троянди"]);
```

5. REST API для отримання товарів (app.py)

Фрагмент серверного коду демонструє маршрут Flask API для отримання списку товарів. Дані зчитуються з бази даних та повертаються клієнтській частині у форматі JSON.

```
@app.route("/api/products", methods=["GET"])
def get_products():
    products = Product.query.all()
    return jsonify([product.to_dict() for product in products])
```

6. REST API для створення замовлення (app.py)

Фрагмент коду відповідає за створення нового замовлення. Сервер приймає дані з форми оформлення, створює запис замовлення та додає до нього товари з кошика.

```
@app.route("/api/orders", methods=["POST"])
def create_order():
    data = request.get_json()

    order = Order(
        name=data.get("name"),
        phone=data.get("phone"),
        email=data.get("email"),
```

```
            address=data.get("address"),
            comment=data.get("comment"),
            payment_method=data.get("paymentMethod"),
            total_price=data.get("totalPrice", 0),
        )

        db.session.add(order)
        db.session.flush()

        for item in data.get("items", []):
            order_item = OrderItem(
                order_id=order.id,
                product_id=item.get("id"),
                title=item.get("title"),
                price=item.get("price"),
                quantity=item.get("quantity"),
                selected_size=item.get("selectedSize"),
            )
            db.session.add(order_item)

        db.session.commit()

        return jsonify({
            "success": True,
            "message": "Замовлення успішно створено",
            "order_id": order.id
        }), 201
```

7. Модель товару бази даних (models.py)

Фрагмент коду показує модель товару, яка використовується для зберігання інформації про продукцію в базі даних. У моделі передбачено назву товару, категорію, ціну, рейтинг, зображення, опис, склад, інформацію про догляд і доставку.

```
class Product(db.Model):
    __tablename__ = "products"

    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(150), nullable=False)
    category = db.Column(db.String(50), nullable=False)
    category_label = db.Column(db.String(100))
    price = db.Column(db.Integer, nullable=False)
    rating = db.Column(db.Float, default=0)
    image = db.Column(db.String(255))
```

```
short_description = db.Column(db.Text)
description = db.Column(db.Text)
composition = db.Column(db.Text)
care_tips = db.Column(db.Text)
delivery_info = db.Column(db.Text)
article = db.Column(db.String(50))
popular = db.Column(db.Boolean, default=False)
```