

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Інтернет-магазин квітів»

Виконав студент групи ІПЗ-21-5

ПАВЛЕНКО Оксана Віталіївна

Керівник роботи:

ВЛАСЕНКО Олег Васильович

Рецензент:

ЧИЖМОТРЯ Олена Геннадіївна

Житомир – 2025

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«26» лютого 2025р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ПАВЛЕНКО Оксана Віталіївна**

Керівник роботи: **ВЛАСЕНКО Олег Васильович**

Тема роботи: **«Інтернет-магазин квітів»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2025р., №61/с**

Вихідні дані для роботи: технічні, функціональні та інформаційні вимоги до розробки вебдодатку для онлайн-продажу квітів із можливістю керування асортиментом, замовленнями та інтеграцією оплати через LiqPay.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Власенко О.В.	27.02.25	27.02.25
2	Власенко О.В.	27.02.25	27.02.25
3	Власенко О.В.	27.02.25	27.02.25

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебдодатку Інтернет-магазину квітів та пояснювальної записки. Пояснювальна записка до випускної роботи загально містить 76 сторінок, з них 52 друкованого тексту та 22 додатків, в документі 41 ілюстрація, 17 таблиць та 31 найменування використаних джерел.

Метою роботи є створення зручного та функціонального онлайн-магазину квітів з використанням сучасних вебтехнологій, який дозволяє користувачам переглядати каталог товарів, формувати власні букети, оформлювати замовлення та здійснювати оплату через систему LiqPay.

Архітектура системи побудована за патерном MVC, що забезпечує ефективну взаємодію між логікою додатку, користувацьким інтерфейсом і базою даних. Адміністративна частина надає можливість керування асортиментом рослин, букетів, категорій і замовлень.

Інтерфейс додатку є інтуїтивно зрозумілим, що дозволяє як звичайним користувачам зручно здійснювати покупки, так і адміністраторам ефективно управляти контентом сайту.

КЛЮЧОВІ СЛОВА: ІНТЕРНЕТ-МАГАЗИН, MVC, REACT, PHP, КОШИК, LIQPAY, БД, ІНТЕРФЕЙС.

					ІПЗ.КР.Б-121.25 - ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	Інтернет-магазин квітів	Лім.	Арк.	Аркуші	
Розроб.		Павленко О. В							
Керівник		Власенко О.В.					3	76	
						ФІКТ, гр. ІПЗ-21-5			
Н. контр.									
Затвердив									

ЗМІСТ

РЕФЕРАТ	3
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	5
ВСТУП	6
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В РОЗРОБЦІ ІНТЕРНЕТ-МАГАЗИНУ КВІТІВ.....	8
1.1. Постановка задачі	8
1.2. Аналіз аналогів програмного продукту	11
1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	16
Висновки до першого розділу.....	18
РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ КВІТІВ	19
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	19
2.2. Розробка системи збереження даних	25
2.3. Проектування та реалізація алгоритмів роботи системи.....	29
2.4 Реалізація функціоналу Інтернет-магазину квітів.....	31
Висновки до другого розділу.....	34
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ-МАГАЗИНОМ КВІТІВ.....	35
3.1. Порядок встановлення та налаштування параметрів платформи.....	35
3.2. Структура інтерфейсу Інтернет-магазину квітів	36
3.3. Тестування програмного продукту	49
Висновки до третього розділу	50
ВИСНОВКИ.....	51
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	52
Додатки.....	54

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ІМ – Інтернет-магазин

API (Application Programming Interface) — це інтерфейс прикладного програмування, тобто набір правил, за якими одна програма може взаємодіяти з іншою.

MVC (Model–View–Controller) — це архітектурний шаблон проєктування програмного забезпечення.

CRUD — це аббревіатура, яка описує чотири основні дії з даними в базі даних або у вебзастосунках:

- C — Create (створення)
- R — Read (читання/перегляд)
- U — Update (оновлення/редагування)
- D — Delete (видалення)

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ВСТУП

Актуальність теми. В наш час продаж через веб-застосунки є одним з найбільш динамічних напрямів розвитку бізнесу. Зростання потреби в онлайн-купівлях зумовлене зручністю та економією часу. Особливу популярність останнім часом отримали Інтернет-магазини, котрі пропонують товари з доставкою, серед яких важливе місце посідають сервіси доставки квітів.

Створення Інтернет-магазину квітів є актуальним з огляду на сезонний попит, наявність особливих подій (свята, весілля, дні народження), а також потребу швидкого реагування на потреби клієнтів у подарунковій продукції. На відміну від традиційних магазинів, онлайн-сервіси забезпечують цілодобову доступність, можливість наперед замовити букет, переглянути його фото та детальний опис, а також обрати інші товари (кімнатні або садові рослини).

З технічної точки зору, розробка подібного веб-застосунку дозволяє реалізувати сучасні підходи до проєктування програмного забезпечення, зокрема модульність, клієнт-серверну архітектуру, інтеграцію з платіжними системами.

Метою випускної кваліфікаційної роботи бакалавра є розробка Інтернет-магазину.

Для створення повноцінного Інтернет-магазину квітів необхідно виконати низку взаємопов'язаних завдань.

На початку слід провести детальний аналіз теми. Це включає дослідження поточних рішень на ринку, зокрема відомих Інтернет-магазинів, що спеціалізуються на продажі квітів. Аналіз такого типу дозволяє визначити сучасні вимоги користувачів, типові функціональні можливості подібних систем, а також недоліки, які варто уникнути під час розробки власного проєкту. Особлива увага має бути приділена зручності користування, асортименту, способам оплати, доставці, фільтрам та сортуванню.

Після цього необхідно обрати відповідні методи проєктування для виконання завдань. Вибір технологій має залежати від надійності, масштабованості, безпеки та зручності подальшого обслуговування системи.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

На цьому етапі відбувається безпосереднє проектування архітектури системи. Частиною цього процесу є створення логічної структури взаємодії між елементами ІМ. Інтерфейс користувача має бути простим для розуміння, красивим і адаптованим до різних типів пристроїв. Розробляється паралельно серверна частина, яка має на меті обробку запитів користувачів, керування базою даних і інтеграцію з зовнішніми сервісами. Налагодження зв'язку з платіжними системами є важливим для безпечних і простих фінансових операцій.

Після завершення основного етапу розробки програмного продукту необхідно провести комплексне тестування. Це дозволяє виявити логічні недоліки, технічні помилки та невідповідності вимогам користувачів. Результати тестування використовуються для доопрацювання системи, вирішення проблем і оптимізації роботи ресурсу. Такий підхід сприяє успішному запуску проекту, підвищує задоволеність клієнтів і гарантує стабільне функціонування ІМ.

Об'єктом дослідження є процес розробки та функціонування Інтернет-магазину, який забезпечує можливість перегляду товарів, оформлення замовлення, вибору способу доставки та здійснення оплати онлайн.

Предмет дослідження є методи та засоби реалізації Інтернет-магазину квітів, зокрема програмні підходи до створення веб-інтерфейсу, організації бази даних товарів, реалізації функціоналу замовлення, оплати та доставки, а також інтеграції з зовнішніми сервісами (платіжна система - LiqPay).

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В РОЗРОБЦІ ІНТЕРНЕТ-МАГАЗИНУ КВІТІВ

1.1. Постановка задачі

Метою кваліфікаційної роботи бакалавра є проектування та розробка Інтернет-магазину квітів (веб-сайту та API), з використанням сучасного підходу до побудови програмних продуктів, який дає можливість масштабування та подальшого вдосконалення магазину.

Сайт має забезпечити зручний для користувача інноваційні можливості купівлі квітів.

Інтернет-магазини — це веб-сайти або мобільні додатки, через які клієнти можуть вибирати, замовляти та оплачувати товари чи послуги в режимі онлайн. Вони є сучасною альтернативою традиційним магазинам і дозволяють вам купувати продукти де завгодно.

Першим ІМ вважається NetMarket, який відкрився 11 серпня 1994 року. Це був один із перших випадків, коли товар (музичний компакт-диск) було продано через інтернет із використанням захищеного з'єднання для онлайн-оплати, The New York Times охарактеризувала це як «...мабуть, першу роздрібну транзакцію в Інтернеті з використанням доступної версії потужного програмного забезпечення для шифрування даних, призначеного для гарантування конфіденційності».

Оплату було здійснено за допомогою безпечного протоколу SSL, що було дуже новаторським на той час. Перший товар, який купили онлайн — це CD з альбомом Стінга "Ten Summoner's Tales". Хоча до цього вже існували електронні системи замовлень через модеми або комп'ютерні мережі (наприклад, CompuServe), саме NetMarket вважається першим сучасним Інтернет-магазином, подібним до тих, які ми знаємо сьогодні.

Основні характеристики Інтернет-магазинів

Однією з головних складових таких платформ є каталог товарів. Він виконує не лише функцію демонстрації асортименту, а й надає користувачам

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

вичерпну інформацію про кожну позицію. У каталозі зазвичай містяться якісні фотографії товарів, детальні описи, актуальні ціни та інше.

Не менш важливою частиною є функціонал кошика та механізм оформлення замовлення. Після вибору товарів користувач може додати їх до кошика, де зберігаються обрані позиції до моменту підтвердження покупки. Далі відбувається процес оформлення замовлення, під час якого клієнт обирає спосіб доставки, зазначає адресу, вводить контактні дані й обирає відповідний варіант оплати.

Переважає більшість сучасних Інтернет-магазинів підтримує онлайн-оплату, що є зручним і швидким способом розрахунку. Користувачі можуть скористатися банківськими картками, інтернет-банкінгом, електронними гаманцями або мобільними платіжними системами, такими як Google Pay, Apple Pay, LiqPay. Також часто доступна можливість оплати накладеним платежем при отриманні товару.

Система доставки відіграє важливу роль у загальному сервісі магазину. Покупцям зазвичай пропонуються різні варіанти: доставка кур'єром до дверей, пересилання через поштові оператори, самовивіз із фізичних пунктів видачі.

Для зручності керування ІМ часто передбачають створення особистого кабінету адміністратора. Це дає змогу переглядати наявні замовлення, перевіряти статус оплати, спосіб доставки та контактну інформацію клієнтів, а також редагувати різноманітні дані про товари та функціонал сайту.

Окрім базової функціональності, Інтернет-магазини можуть також пропонувати додаткові можливості, які покликані зробити користувацький досвід приємнішим і мотивувати до повторних покупок. Це можуть бути бонусні програми, система знижок, персональні рекомендації, інтерактивні елементи, наприклад, віртуальний букет або конструктор подарункових наборів. Такі опції не лише покращують імідж бренду, а й дозволяють виділитися серед конкурентів на ринку.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

В результаті повинно бути створено веб-сайт, за допомогою якого будь-який бажаючий може замовити букет квітів або іншу рослинність для оселі або саду з можливістю самовивозу або доставки до місця замовлення.

Розробка платформи для Інтернет-магазину включає низку важливих етапів, кожен з яких виконує ключову роль у формуванні повноцінного, стабільного та зручного у використанні програмного продукту.

Початковим етапом є побудова архітектури серверної частини додатку. На цьому етапі визначаються основні принципи взаємодії між компонентами бекенду, обираються технології, що забезпечуватимуть ефективну обробку даних, високу продуктивність і надійність системи. Архітектура має бути масштабованою та адаптованою до можливого зростання навантаження, а також відповідати вимогам безпеки.

Після формування архітектурного плану розпочинається безпосередня розробка функціональної частини серверної логіки. Важливим завданням цього етапу є створення віддаленої бази даних, яка слугуватиме основним сховищем для всієї інформації, зокрема даних про товари, замовлення, клієнтів, оплату та доставку. Також розробляється програмний інтерфейс взаємодії — API, що забезпечує зв'язок між клієнтською частиною та сервером. Саме через API відбувається обмін даними, що дозволяє реалізувати динамічну взаємодію між користувачем і системою. Крім того, невід'ємним елементом серверної частини є інтеграція з платіжними сервісами, що дозволяє обробляти онлайн-оплати у безпечний та зручний спосіб.

Наступним кроком у процесі розробки є створення дизайну вебсайту. Цей етап починається з формування загальної візуальної концепції інтерфейсу, яка має відображати стиль бренду, бути привабливою та зрозумілою для користувача. Після цього відбувається створення UI-компонентів — окремих елементів інтерфейсу, таких як кнопки, поля вводу, банери, тощо. Завершальним підетапом дизайну є розробка повноцінних макетів сторінок, включаючи головну сторінку, каталог, картку товару, сторінку замовлення, кошик та інші функціональні частини платформи.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк. 10
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		

Після опрацювання дизайну розпочинається побудова архітектури клієнтської частини додатку. Цей процес передбачає реалізацію логіки інтерфейсу користувача, забезпечення швидкого та коректного відображення вмісту сторінок, а також налаштування взаємодії з API. У результаті користувач має можливість отримувати інформацію з сервера, оформляти замовлення, здійснювати оплату тощо.

Завершальним етапом є тестування додатку, що включає як функціональні, так і візуальні перевірки. Функціональне тестування дозволяє переконатися, що всі основні можливості системи працюють відповідно до технічного завдання — від авторизації користувача до обробки замовлення та оплати. Тестування графічного інтерфейсу, у свою чергу, дає змогу виявити недоліки у візуальному відображенні сайту, оцінити адаптивність до різних екранів і загальну зручність користування. Виявлені помилки усуваються до запуску, щоб забезпечити стабільну роботу та позитивний користувацький досвід.

1.2. Аналіз аналогів програмного продукту

Розробка платформи для інтернет-продажів потребує глибокого аналізу та дослідження конкурентів на ринку. Оскільки онлайн-магазини досить стрімко розвиваються і щодня їх з'являється все більше і більше, варто провести детальний аналіз наявних популярних Інтернет-магазинів для продажу квітів.

Слід зосередитись не лише на зовнішньому вигляді та зручності користування конкурентних сайтів, а й на технічних аспектах їхньої реалізації. Проаналізувати структуру баз даних, принципи побудови бекенду, способи оптимізації продуктивності та механізми безпеки. Це дозволить чітко визначити, які технічні рішення варто запозичити, а які — удосконалити або замінити на ефективніші.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

Одним з найпопулярніших аналогів є Flowers.ua.

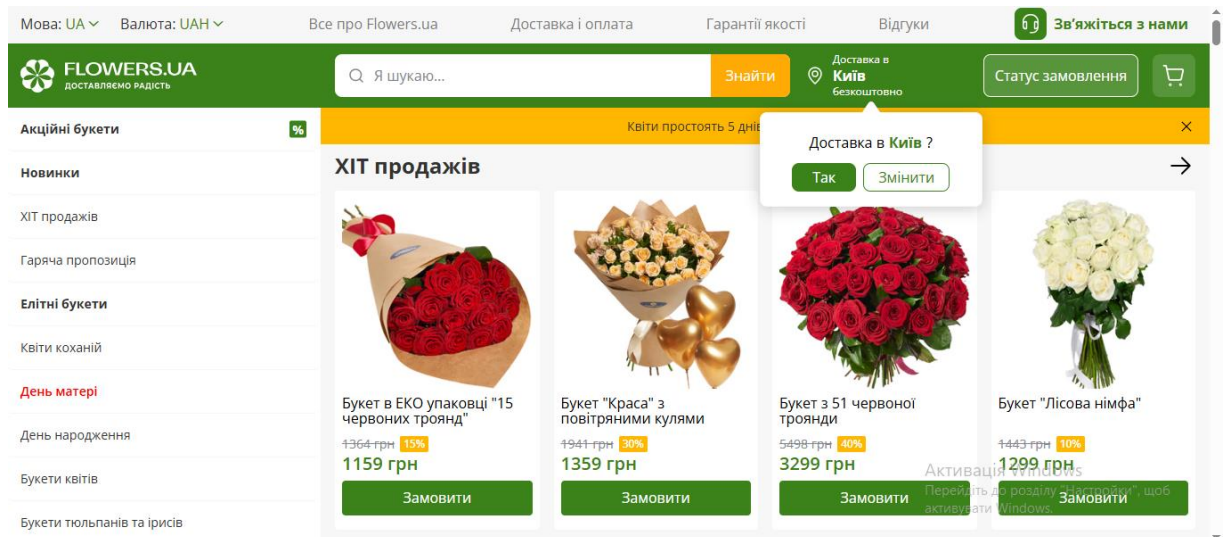


Рис. 1.1. Інтернет-магазин квітів Flowers.ua

Flowers.ua – це національна служба доставки квітів і подарунків по Україні та світу, що на ринку вже більше 17 років. Платформа пропонує дуже широкий вибір асортименту квітів, різноманітне пакування букетів та подарунки, що доповняють привітання. Платформа користується шаленою популярністю через швидкість опрацювання і доставки замовлень, доставку замовлень по всій території України та за кордон, а також через унікальну можливість створення букету за індивідуальним замовленням.

Перевагами є зручний і зрозумілий інтерфейс, а також наявність широкого вибору категорій для замовлення букетів. Користувач може легко знайти потрібний варіант за типом квітів (як монобукети, так і збірні композиції), за адресатом (наприклад, для мами, дитини чи чоловіка), за подією (день народження, весілля тощо) або за ціновою категорією. Важливою перевагою є також інтеграція з платіжними системами та службами доставки завдяки використанню API.

Серед недоліків можна виокремити застарілий дизайн сайту, який потребує оновлення для відповідності сучасним візуальним та функціональним стандартам.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Наступний аналог – UFL.ua.

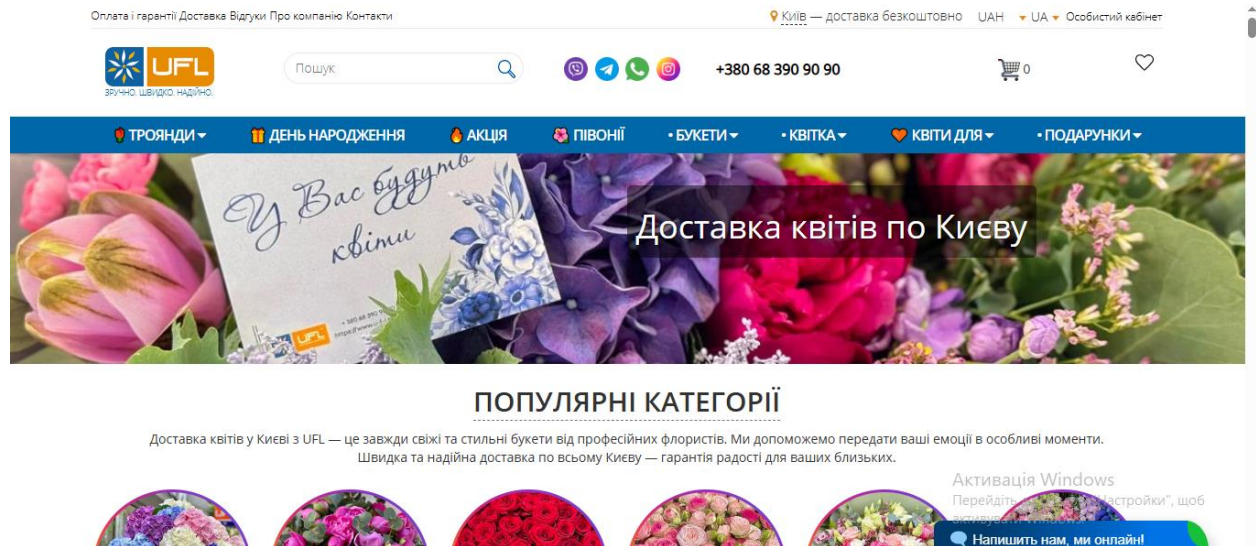


Рис. 1.2. Інтернет-магазин квітів UFL.ua

UFL - це велика квіткова мережа, що працює з 2003 року на ринку доставки букетів та подарунків. Ця платформа відома завдяки численним позитивним відгукам клієнтів, адже фото на сайті відповідають дійсності. Понад 650 тис. успішно виконаних замовлень зробили компанію лідером на ринку квіткових поздоровлень у 2023 р.

Серед переваг можна відзначити інтеграцію з платіжними системами та службами доставки за допомогою API, що забезпечує зручність оформлення замовлення. Також позитивним є наявність лайв-чату для оперативної підтримки користувачів та використання анімацій, які роблять сайт більш динамічним і привабливим.

Недоліками є перевантажений інтерфейс, що може ускладнювати навігацію, а також складна структура меню, яка не завжди інтуїтивно зрозуміла для відвідувачів.

Слідуючий аналог – FloraQueen.

FloraQueen — це іспанська фірма, заснована у 2004 році в Барселоні, яка спеціалізується на міжнародній доставці квітів і подарунків. Вона працює більш ніж у 100 країнах світу, зокрема й в Україні. Компанія активно розвиває онлайн-платформу, яка дозволяє швидко обрати букет, оформити замовлення та надіслати його за будь-якою адресою в декілька кліків.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						13

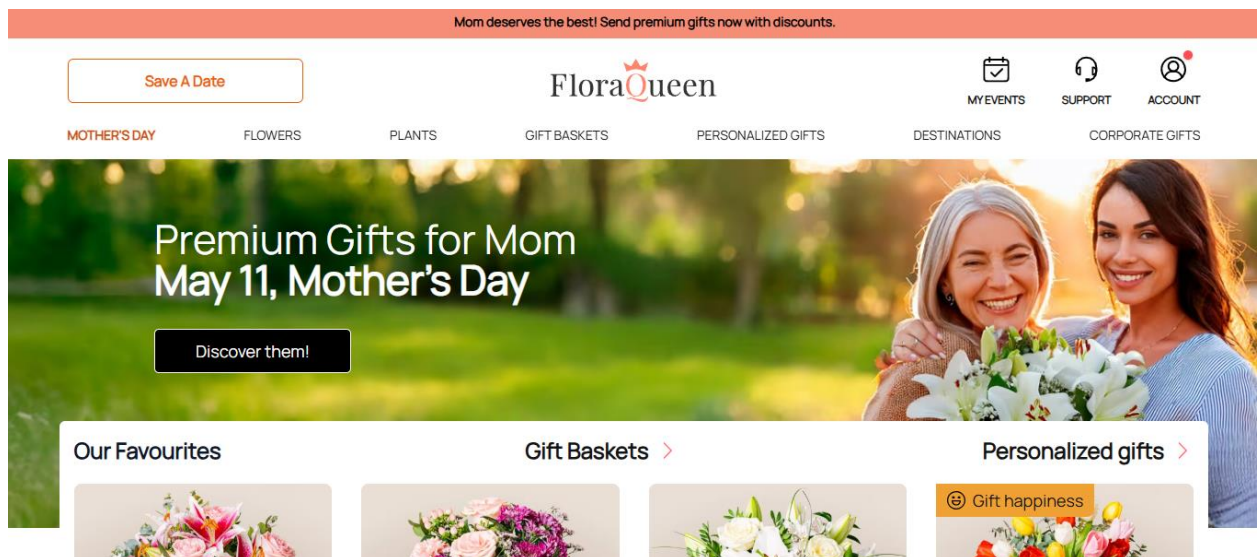


Рис. 1.3. Інтернет-магазин квітів FloraQueen

До переваг належать інтуїтивно зрозумілий інтерфейс і зручна навігація, що дозволяє користувачам швидко орієнтуватися на сайті. Система автоматично визначає геолокацію, що спрощує процес замовлення, а також забезпечує безпечні платежі, що підвищує довіру користувачів.

Водночас існують і деякі недоліки. Іноді система некоректно визначає мову або валюту, що може спричинити незручності для користувачів. Крім того, сторінки сайту можуть завантажуватись повільно, що негативно впливає на загальне враження від користування ресурсом.

Варто згадати ще один аналог – MyGlobalFlowers.

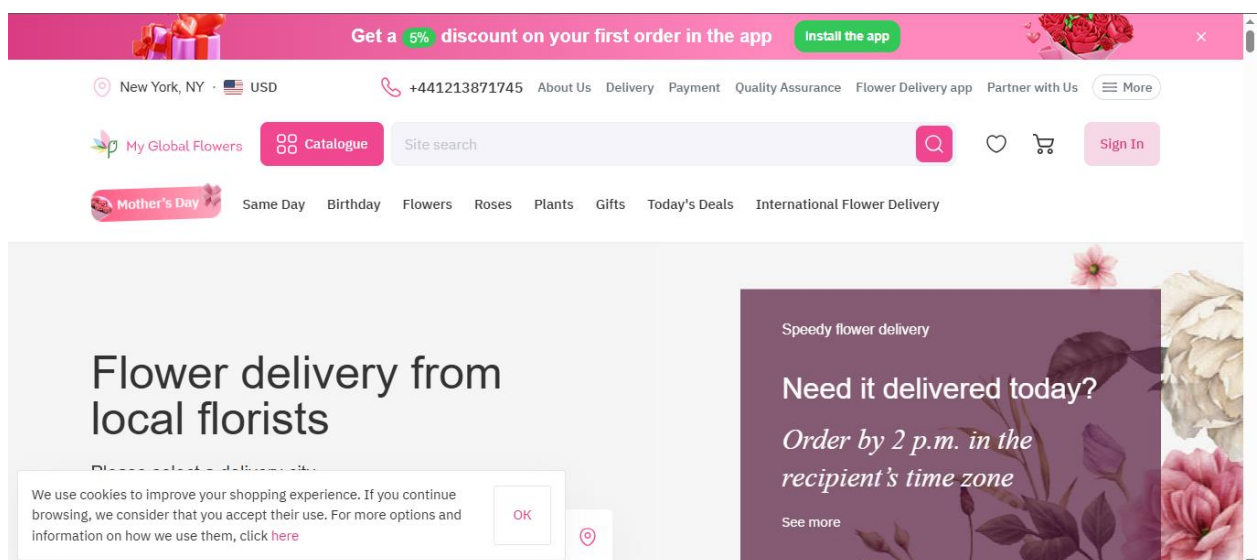


Рис. 1.4. Інтернет-магазин квітів MyGlobalFlowers

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк. 14
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		

MyGlobalFlowers — це порівняно молодий, але активно зростаючий онлайн-сервіс доставки квітів, який фокусується саме на міжнародному ринку. Компанія співпрацює з локальними флористами у понад 100 країнах, що дозволяє доставляти квіти навіть у віддалені регіони.

MyGlobalFlowers робить ставку на простоту: зрозумілий сайт, швидка навігація, легкий процес покупки. Компанія також прагне забезпечити високу якість сервісу, контролюючи виконання замовлень і пропонуючи гарантії повернення грошей у разі невдоволення клієнта.

Серед переваг варто виділити зручний і мінімалістичний дизайн, який не перевантажує користувача та сприяє приємному візуальному сприйняттю. Крім того, на сайті реалізовано зрозумілу систему пошуку й ефективні фільтри, що полегшують вибір товару.

Проте є й недоліки — на сайті бракує інтерактивних елементів, що може зробити взаємодію менш захопливою та зменшити зацікавленість користувача, та відсутня можливість створення букету за власними запитами.

Проаналізувавши усі вищезгадані платформи можна прослідкувати декілька спільних мінусів і плюсів.

Таблиця 1.1

Таблиця порівняння Інтернет-магазинів квітів

Критерій	Flowers.ua	UFL.ua	FloraQueen	MyGlobalFlowers
Інтеграція з платіжними системами	+	+	+	+
Зручність інтерфейсу	+	±	+	+
Широкий вибір асортименту	+	+	+	±
Автоматичне визначення геолокації	—	—	+	—
Створення індивідуального букету	+	+	—	—
Актуальний сучасний дизайн	—	—	+	+
Швидкість роботи сайту	+	+	±	+

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

На основі проведеного аналізу можна зробити висновок, що сучасні Інтернет-магазини квітів орієнтовані на зручність для користувача. Однак більшість із них мають проблеми з дизайном та навігацією. Тому під час розробки власного програмного продукту доцільно поєднати сильні сторони конкурентів (швидкість, зручність, інтеграції) та уникнути їхніх недоліків, особливо в частині UI/UX і гнучкості функціоналу.

1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

У процесі розробки інтернет-магазину квітів важливим етапом є вибір інструментальних засобів, які забезпечують не лише ефективність, а й стабільність, масштабованість та зручність подальшої підтримки. Обраний стек РНР для бекенд-розробки та React для фронтенду має свої очевидні переваги, але для більш обґрунтованого пояснення «Чому так?» варто розглянути й альтернативні підходи з використанням інших мов програмування і фреймворків, аби краще зрозуміти переваги та обмеження.

Якщо порівняти РНР із сучасними серверними технологіями, наприклад, з Node.js, варто врахувати, що Node.js, який базується на JavaScript, пропонує однорідний стек мов для фронтенду і бекенду. Це дозволяє розробникам працювати в одному мовному середовищі, що полегшує підтримку і спрощує комунікацію між командами. Node.js має асинхронну модель обробки подій, що підвищує продуктивність при роботі з великою кількістю одночасних з'єднань, що корисно для високонавантажених проєктів. Проте такий підхід вимагає більшого залучення фізичних ресурсів для подальшої підтримки проєкту. РНР, у свою чергу, має більш лінійний і зрозумілий підхід, особливо у поєднанні з патерном MVC, що робить його привабливим для швидкої розробки класичних веб-застосунків.

Ще одним популярним бекенд-фреймворком є Django на базі мови Python. Django пропонує високу ступінь автоматизації, вбудовані засоби безпеки, а також "батарейки в комплекті" — готові рішення для більшості типових завдань, що значно прискорює розробку складних систем. Python у цілому є дуже

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

популярною мовою з широкою екосистемою, що охоплює не лише веб-розробку, а й аналіз даних, машинне навчання, автоматизацію тощо. Однак для реалізації SPA (Single Page Application) із складним клієнтським інтерфейсом у Django часто використовують додаткові інструменти для фронтенду, подібно до React, що може ускладнювати архітектуру, крім цього React сам по собі дуже повільно оброблює інформацію, що може уповільнити роботу веб застосунку. У порівнянні, PHP безпосередньо інтегрується з HTML і дозволяє швидко будувати класичні веб-сторінки, а з сучасними підходами — створювати RESTful API для React.

Щодо фронтенд-фреймворків, React виділяється завдяки компонентній архітектурі та віртуальному DOM, що забезпечує високу продуктивність. Проте варто розглянути й альтернативи, наприклад, Angular або Vue.js. Angular є потужним фреймворком з повним набором інструментів «з коробки» — він включає маршрутизацію, форму валідації, роботу з HTTP, тощо. Це дає змогу будувати великі корпоративні застосунки з уніфікованою структурою, але разом із тим Angular має більш круту криву навчання через складність і велике охоплення функціоналу. Vue.js, навпаки, пропонує більш легкий і гнучкий підхід, він швидко освоюється, має легкий синтаксис і водночас потужний функціонал, що робить його привабливим для проєктів середнього масштабу. React посередині за рівнем складності і при цьому має найбільшу екосистему та підтримку, що важливо для подальшого розвитку проєкту.

Також варто згадати, що вибір інструментів нерідко залежить від специфіки проєкту, команди розробників і майбутніх планів на підтримку. PHP та React створюють збалансований тандем, що дозволяє швидко вивести на ринок якісний продукт з чітким розділенням обов'язків між бекендом і фронтендом. Водночас більш складні фреймворки, такі як Django чи Angular, можуть бути перевагою для великих корпоративних систем з високими вимогами до безпеки та уніфікованої архітектури.

Щодо вимог до апаратного забезпечення, варто відзначити, що і PHP, і React, на відміну від більш важких фреймворків і платформ, не пред'являють

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						17

надмірних ресурсних вимог. Сервер на базі PHP є відносно легким у налаштуванні і не потребує потужних процесорів або великої кількості оперативної пам'яті, що знижує загальні витрати на хостинг. У той же час Node.js, через свій асинхронний і подієвий характер, може ефективніше використовувати ресурси при великому навантаженні, але потребує більш ретельного налаштування середовища. Фронтенд на React також оптимізований для роботи на клієнтських пристроях з різним рівнем продуктивності, що дає змогу підтримувати комфортний користувацький досвід навіть на пристроях середнього рівня.

Отже, вибір PHP і React в контексті розробки інтернет-магазину квітів є результатом компромісу між простотою впровадження, продуктивністю, підтримкою спільноти та гнучкістю архітектури. Цей стек технологій відповідає сучасним вимогам і водночас дає змогу забезпечити надійність і масштабованість платформи. Водночас альтернативи, такі як Node.js або Django на бекенді, а також Angular чи Vue.js на фронтенді, можуть бути розглянуті в залежності від особливостей бізнесу, складності функціоналу та досвіду команди розробників, що дозволить максимально адаптувати платформу під конкретні потреби проекту.

Висновки до першого розділу

Проведено аналіз предметної області, за допомогою якого були визначені основні вимоги до Інтернет-магазину квітів.

В процесі дослідження були проаналізовані та порівняні різні платформи аналогічного типу.

Крім цього, були сформовані основні вимоги для власної такої платформи. На основі відповідних вимог та критеріїв був сформований набір основних технологій і засобів реалізації для розробки вебдодатку.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

РОЗДІЛ 2. ПРОЕКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ КВІТІВ

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

Програмний продукт належить до категорії електронної комерції, тож основними акторами в системі виступають користувачі та адміністратор. Кожен з них взаємодіє із системою відповідно до своєї ролі та ініціює ключові дії у межах функціональних можливостей магазину. Користувач може переглядати товари та додавати їх до кошика, оформлювати замовлення та здійснювати оплату, а адміністратор — керувати асортиментом, замовленнями та користувачами.

До основних варіантів використання належать перегляд каталогу товарів, пошук і фільтрація букетів, додавання товарів до кошика, оформлення замовлення, здійснення онлайн-оплати, а також керування товарами та обробка замовлень з боку адміністратора.

З метою покращення розуміння логіки роботи системи, її структури та ролей користувачів, було створено діаграму варіантів використання (Use Case Diagram) (рис. 2.1). Ця діаграма демонструє основні сценарії взаємодії між користувачами та функціоналом системи. Кожен варіант використання відображає окрему функціональність Інтернет-магазину, яку можна ініціювати відповідним актором, і наочно ілюструє можливості взаємодії з платформою.

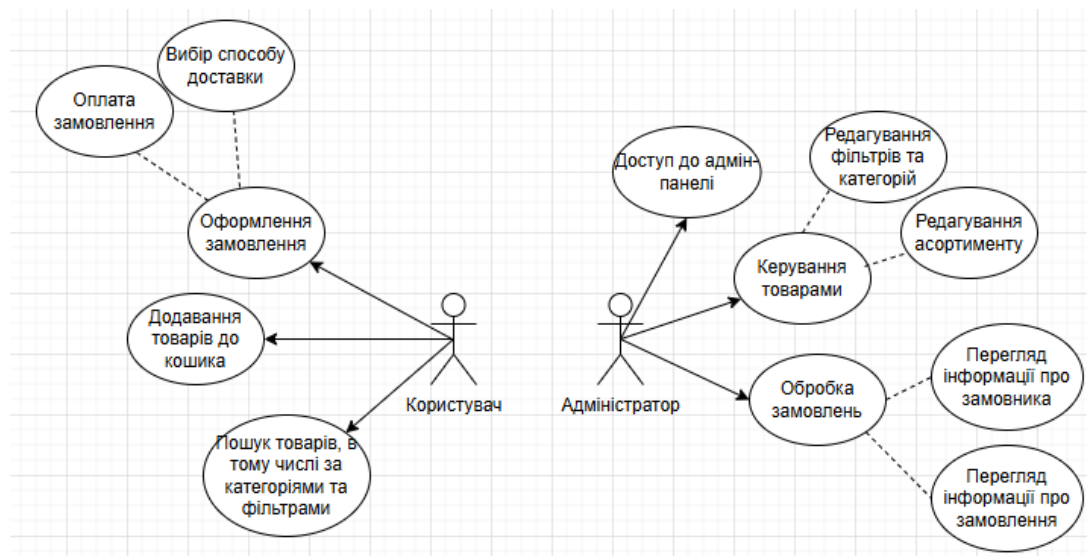


Рис. 2.1. Діаграма варіантів використання

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

Опис діаграми варіантів використання наведено в таблицях 2.1 – 2.6

Таблиця 2.1

Опис прецеденту «Пошук товарів»

Назва варіанту використання	Пошук товарів
Основні актори	Користувач
Короткий опис	Можливість перегляду вмісту платформи, а також здійснення пошуку, включаючи доступні категорії та фільтри

Таблиця 2.2

Опис прецеденту «Додавання товарів до кошику»

Назва варіанту використання	Додавання товарів до кошику
Основні актори	Користувач
Короткий опис	Зберігання товару в кошику для подальших операцій з ним

Таблиця 2.3

Опис прецеденту «Оформлення замовлення»

Назва варіанту використання	Оформлення замовлення
Основні актори	Користувач

Продовження таблиці 2.3

Короткий опис	Доданий у кошик товар, можна замовити шляхом оплати та вибору способу доставки
---------------	--

Таблиця 2.4

Опис прецеденту «Керування товарами»

Назва варіанту використання	Керування товарами
Основні актори	Адміністратор
Короткий опис	Редагування товарів (додати, видалити чи редагувати товар - фото, опис, ціна і т.п.), редагування іншої інформації (додавання чи видалення фільтрів та категорій)

Таблиця 2.5

Опис прецеденту «Обробка замовлень»

Назва варіанту використання	Обробка замовлень
Основні актори	Адміністратор
Короткий опис	Відображення у базі даних інформації про замовника, статус оплати, спосіб доставки та саме замовлення

Таблиця 2.6

Опис прецеденту «Доступ до адмін-панелі»

Назва варіанту використання	Доступ до адмін-панелі
Основні актори	Адміністратор
Короткий опис	Повний доступ до інформації про користувачів та їх замовлень та можливість редагування тієї чи іншої інформації на платформі

Основна ціль розробки сайту Інтернет-магазину квітів полягає у створенні середовища, де кожен бажаючий зможе замовити собі букет, кімнатну чи садову рослину. Зі зручними методами оплати, сортування та можливістю зібрати свій власний букет з квітів наявних на сайті.

Основна мета розробки полягає у створенні зручного онлайн-середовища, яке дозволить користувачам легко знаходити потрібні товари, отримувати актуальну інформацію про наявність квітів та рослин, а також оформлювати замовлення у будь-який зручний час. Сайт має на меті зробити процес вибору та покупки квітів максимально простим, інтуїтивним і приємним для кожного відвідувача.

Вимоги до користувачів платформи поділяються на дві основні категорії - для адміністратора та для клієнта.

Внутрішній користувач, який виконує роль адміністратора, повинен мати можливість повного керування вмістом Інтернет-магазину. Це передбачає можливість додавання, редагування та видалення інформації про магазини, управління асортиментом продукції — включно з рослинами та букетами.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

Адміністратор також повинен мати змогу створювати нові категорії та типи рослин, а також змінювати або видаляти вже наявні. Не менш важливою функцією є доступ до перегляду та аналізу замовлень, що надходять через систему, оскільки це дозволяє контролювати стан продажів і забезпечувати належний рівень обслуговування.

Зі свого боку, клієнт, як зовнішній користувач, очікує на зручну та інтуїтивно зрозумілу взаємодію з сайтом. Основними функціональними можливостями для нього є перегляд сторінок магазину, ознайомлення з асортиментом продукції, додавання товарів до кошика, редагування його вмісту та зручна оплата замовлень. Для завершення транзакцій клієнт повинен мати змогу оплатити замовлення через інтегровану платіжну систему, наприклад, LiqPay, що забезпечує швидкість та безпеку платежів.

Функціональні вимоги до вебплатформи включають цілу низку критично важливих елементів, які забезпечують її ефективну роботу.

Основою системи є серверна частина, яка повинна функціонувати у форматі API, що дозволить організувати чітку взаємодію між клієнтським інтерфейсом і сервером. Однією з ключових функцій є реалізація можливості онлайн-оплати замовлень за допомогою популярного платіжного сервісу LiqPay. Це значно підвищує зручність та безпеку користувачів при здійсненні транзакцій. Важливо також, щоби доступ до товарів на сайті був зручним, логічно структурованим і інтуїтивно зрозумілим, що дозволить покупцям швидко знаходити потрібну продукцію. Не менш важливою частиною функціоналу є наявність кошика, який повинен бути простим у використанні, дозволяючи додавати, редагувати або видаляти товари.

Адміністративна панель має бути розроблена таким чином, щоб адміністратору було легко редагувати інформацію на сайті, додавати нові позиції, переглядати замовлення та керувати контентом. Вся інформація повинна зберігатися у внутрішній базі даних, яка забезпечить надійне збереження та швидке опрацювання даних.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

Нефункціональні вимоги зосереджені на користувацькому досвіді, надійності системи та її здатності обробляти навантаження. Очікується, що користувачі зможуть швидко освоїти вебсайт — новачкам це має займати не більше 3–5 хвилин, тоді як досвідчені користувачі зможуть орієнтуватися в інтерфейсі менш ніж за 2 хвилини. При цьому всі дії на сайті повинні виконуватись оперативно, а час реакції інтерфейсу не повинен перевищувати трьох секунд. Інтерфейс сайту повинен бути інтуїтивно зрозумілим і зручним, щоб користувачам не потрібно було додатково навчатися або витратити час на розбір функцій. У плані надійності платформа має бути доступною цілодобово, без перерв у роботі чи збоїв у системі. Крім того, вона повинна забезпечувати високу продуктивність — це означає, що навіть при великій кількості одночасних відвідувачів, система має залишатися стабільною, швидкою та повністю функціональною.

Аналіз функціональних вимог дозволив виділити наступні сутності, що забезпечать програмну реалізацію додатку. Діаграму класів бекенду описано на рисунку 2.2.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

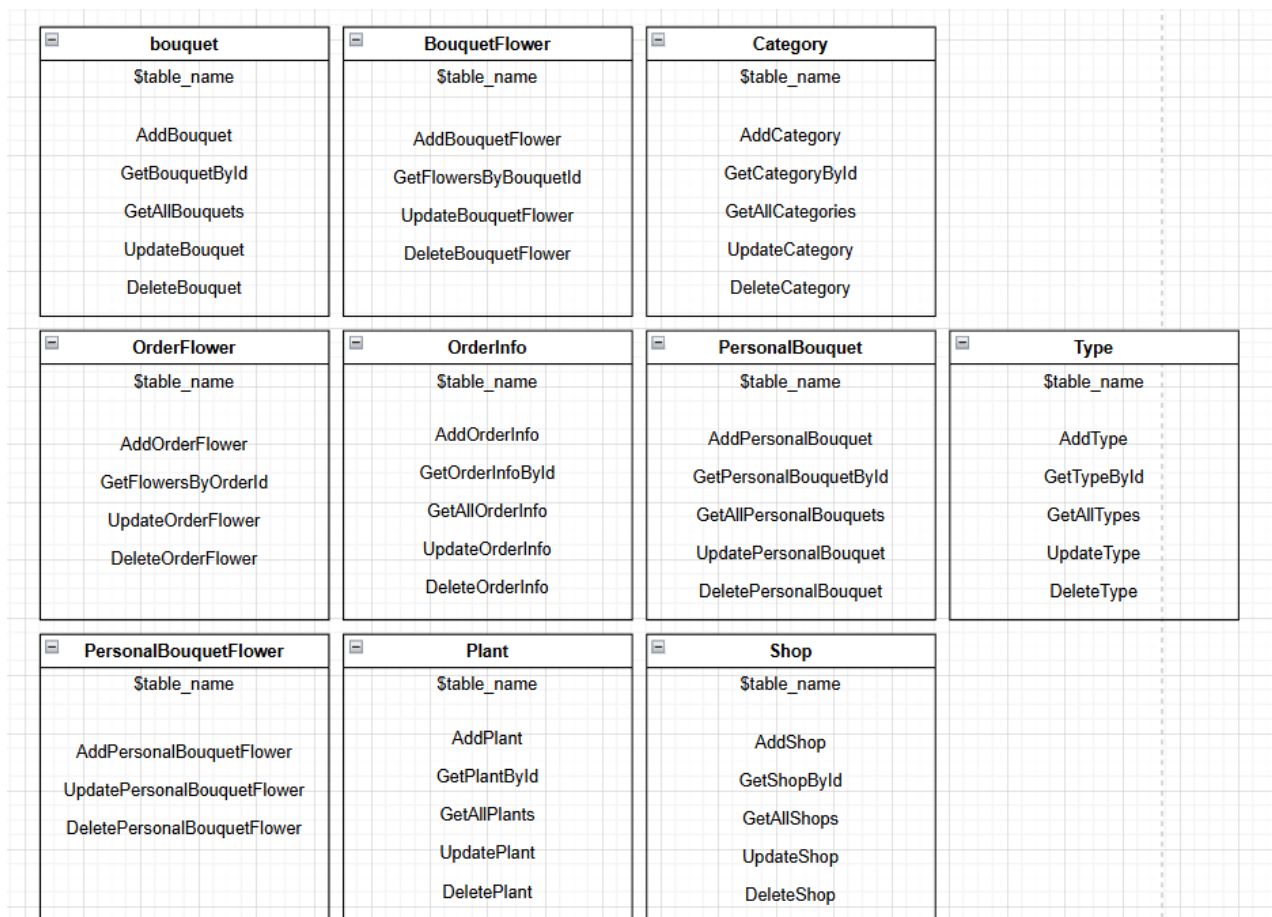


Рис. 2.2. Діаграма класів

На рисунку показані класи моделей, кожна з них відповідає за свою таблицю в базі даних, формально вони кожна з них має CRUD операції та деякі свої утиліті функції для коректної роботи.

Адмін панель спроектована за патерном MVC.

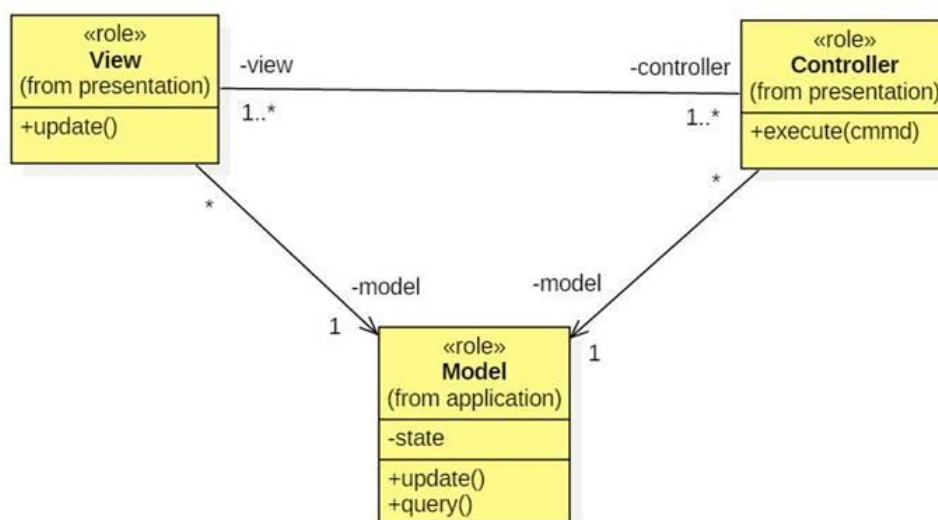


Рис. 2.3. Діаграма MVC

Model відіграє ключову роль у зберіганні й обробці всієї інформації, пов'язаної з товарами, букетами, категоріями, типами рослин, замовленнями та магазинами, включаючи їхню інформацію. Наприклад, коли адміністратор додає нову рослину до системи, ця дія безпосередньо змінює модель, оскільки в базу даних додається новий запис.

View слугує для виведення цієї інформації адміністратору у вигляді таблиць, форм, списків і кнопок, забезпечуючи зручний графічний інтерфейс. Після внесення змін у модель, наприклад, додавання нового букету, представлення автоматично оновлюється й відображає актуальну інформацію на сторінці.

Controller виконує роль посередника між інтерфейсом і логікою програми. Коли адміністратор здійснює певну дію, наприклад, натискає кнопку «Редагувати категорію» або «Додати рослину», контролер отримує цей запит, обробляє його, звертається до відповідного методу моделі, після чого результат змін передається назад до View для оновлення відображення.

Уся система працює злагоджено: контролер взаємодіє з моделлю, змінюючи її стан, модель повідомляє представлення про зміни, а представлення динамічно оновлюється. Такий підхід дозволяє чітко розділити логіку, дані та інтерфейс, що суттєво спрощує розробку, тестування та подальше масштабування адмін-панелі Інтернет-магазину.

2.2. Розробка системи збереження даних

Для зберігання даних у додатку використовується система керування базами даних MySQL, яка вирізняється надійністю, масштабованістю та ефективністю. Вона дозволяє структуровано зберігати інформацію про квіти, букети, замовлення, власні букети, категорії, типи та інше. Завдяки підтримці SQL-запитів розробник має змогу зручно виконувати операції пошуку, оновлення чи видалення інформації. Крім того, MySQL легко інтегрується з популярними мовами програмування, такими як PHP, Python та JavaScript, що забезпечує її широке використання як у веб, так і в десктопних проєктах.

Розглянемо наявні в системі таблиці.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

Таблиця 2.7

Опис таблиці bouquet

Поле	Тип	Призначення
id	int(11)	Ідентифікатор
image	varchar(255)	Картинка букету
name	varchar(55)	Назва букету
price	float	Ціна товару

Таблиця 2.8

Опис таблиці bouquet_flower

Поле	Тип	Призначення
bouquet_id	int(11)	Ідентифікатор букету
plant_id	int(11)	Ідентифікатор рослини
count	int(11)	Кількість рослин в букеті
id	int(11)	Ідентифікатор

Таблиця 2.9

Опис таблиці category

Поле	Тип	Призначення
id	int(11)	Ідентифікатор
name	varchar(55)	Назва категорії

Таблиця 2.10

Опис таблиці order_flowers

Поле	Тип	Призначення
order_id	int(11)	Ідентифікатор замовлення
plant_id	int(11)	Ідентифікатор рослини
bouquet_id	int(11)	Ідентифікатор букету
personal_bouquet_id	int(11)	Ідентифікатор персонального букету
count	int(11)	Кількість в замовленні
id	int(11)	Ідентифікатор

Таблиця 2.11

Опис таблиці type

Поле	Тип	Призначення
id	int(11)	Ідентифікатор
name	varchar(55)	Назва типу

Таблиця 2.12

Опис таблиці shop

Поле	Тип	Призначення
id	int(11)	Ідентифікатор
address	varchar(55)	Адреса магазину

Таблиця 2.13

Опис таблиці order_info

Поле	Тип	Призначення
id	int(11)	Ідентифікатор
date	date	Дата замовлення
phone	varchar(55)	Номер телефону замовника
email	varchar(55)	Пошта замовника
is_delivery	tinyint(1)	Чи вибрано доставку
delivery_address	int(11)	Якщо обрано доставку куди доставляти
delivery_datetime	datetime	Якщо обрано доставку коли доставляти
price	float	Ціна
shop_id	int(11)	Якщо доставку не обрано то з якого магазину забирати

Таблиця 2.14

Опис таблиці personal_bouquet

Поле	Тип	Призначення
id	int(11)	Ідентифікатор

Таблиця 2.15

Опис таблиці personal_bouquet_flower

Поле	Тип	Призначення
id	int(11)	Ідентифікатор
personal_bouquet_id	int(11)	Ідентифікатор персонального букету
plant_id	int(11)	Ідентифікатор рослини
count	int(11)	Кількість

Таблиця 2.16

Опис таблиці plant

Поле	Тип	Призначення
id	int(11)	Ідентифікатор
category_id	int(11)	Ідентифікатор категорії
type_id	int(11)	Ідентифікатор типу
is_bouquet	tinyint(1)	Чи доступна ця рослина в створенні букетів
name	varchar(255)	Назва рослини
image	varchar(255)	Зображення рослини
height	float	Висота рослини
price	float	Ціна рослини

У базі даних системи Інтернет-магазину квітів реалізовано логічну та структуровану модель збереження інформації, яка охоплює всі ключові аспекти роботи магазину — від каталогізації товарів до обробки замовлень. Основу структури становлять таблиці plant, bouquet та personal_bouquet, які зберігають інформацію про окремі рослини, готові букети та персоналізовані композиції відповідно. Додаткові таблиці, як-от bouquet_flower та personal_bouquet_flower, реалізують зв'язок між букетами та рослинами, вказуючи, які саме квіти входять до складу кожного букету та в якій кількості.

Також у базі даних передбачено таблиці для зберігання інформації про замовлення (order_info, order_flowers), що дозволяє фіксувати деталі замовлень, включаючи вибір доставки, контактні дані клієнта та вміст кожного замовлення. Допоміжні таблиці category, type та shop забезпечують гнучку класифікацію

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						28

рослин за категоріями й типами, а також зберігають дані про фізичні точки продажу. Така структура бази даних дає змогу ефективно управляти асортиментом, обробляти замовлення та надавати користувачам можливість створення власних букетів. Саму діаграму БД можна переглянути нижче.

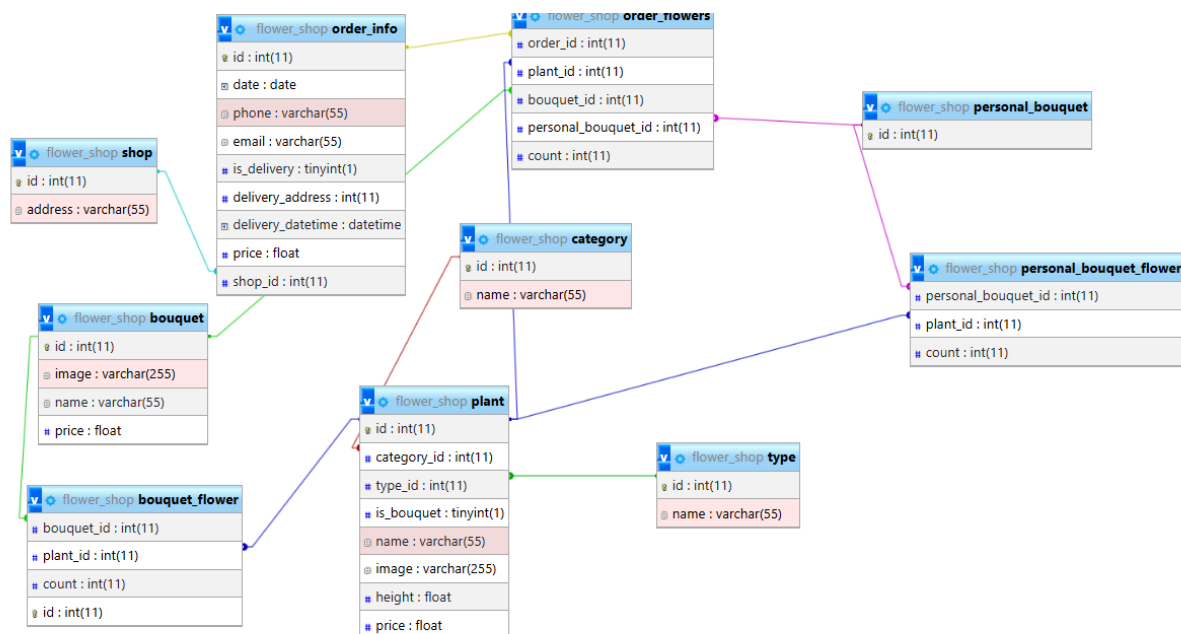


Рис. 2.4. Діаграма бази даних

2.3. Проектування та реалізація алгоритмів роботи системи

Система вирізняється своєю простотою у використанні та містить усі базові функції, необхідні для повноцінного онлайн-магазину квітів. Це забезпечує інтуїтивно зрозумілу взаємодію користувача з усіма елементами інтерфейсу. Користувач легко може знайти потрібні квіти або букети за допомогою зручного пошуку, а також має змогу самостійно сформувати власний букет, вибираючи окремі рослини відповідно до власних уподобань.

Інтерфейс передбачає комфортне керування кошиком — користувач може переглядати обрані товари, редагувати склад кошика, видаляти або додавати нові позиції перед оформленням замовлення. Після остаточного вибору товарів, система пропонує швидкий та безпечний спосіб оплати через платіжний сервіс LiqPay.

Усі етапи взаємодії з платформою, починаючи від відкриття головної сторінки й завершуючи оплатою замовлення, логічно пов’язані між собою та зображені на рисунку 2.5.

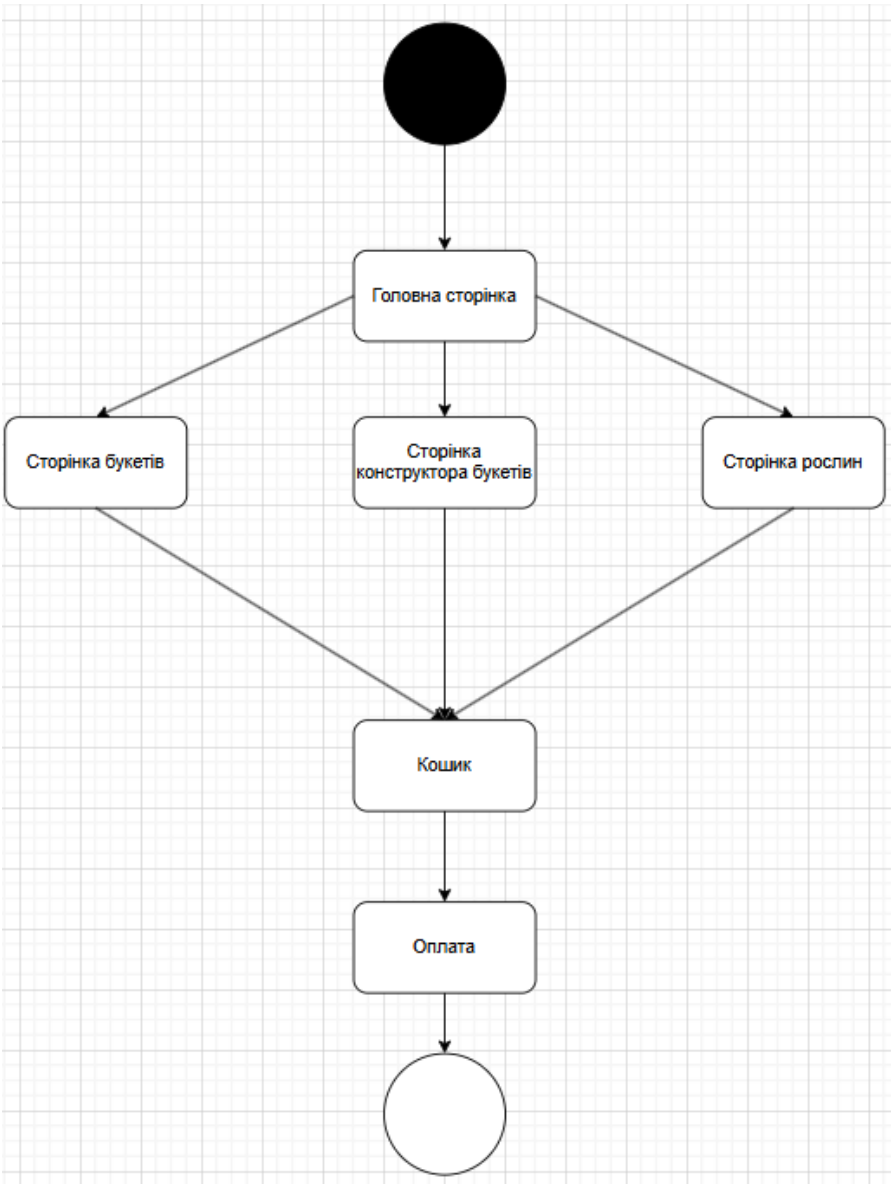


Рис. 2.5. Діаграма активності

Для адміністратора інтерфейс додатку організовано таким чином, щоб забезпечити повний контроль над усім вмістом сайту. Робота починається з дашборду, який відображає основну інформацію про поточний стан магазину, статистику замовлень і загальну динаміку. Звідси адміністратор має змогу перейти до керування магазинами: створювати нові записи, змінювати дані про наявні торгові точки або видаляти ті, що більше не функціонують. Крім цього, у розпорядженні адміністратора є можливість редагувати типи продукції, які

систематизують рослини за характерними ознаками — нові типи можна додати, існуючі — оновити або видалити. Аналогічні дії доступні і для категорій, які ще більше деталізують структуру товарів. Далі адміністратор переходить до управління рослинами: система дозволяє додавати нові позиції до бази, вносити зміни до характеристик або повністю видаляти їх у разі потреби. Окремий розділ стосується букетів — адміністратор має змогу створювати нові композиції, оновлювати інформацію про вже наявні або видаляти застарілі зразки.

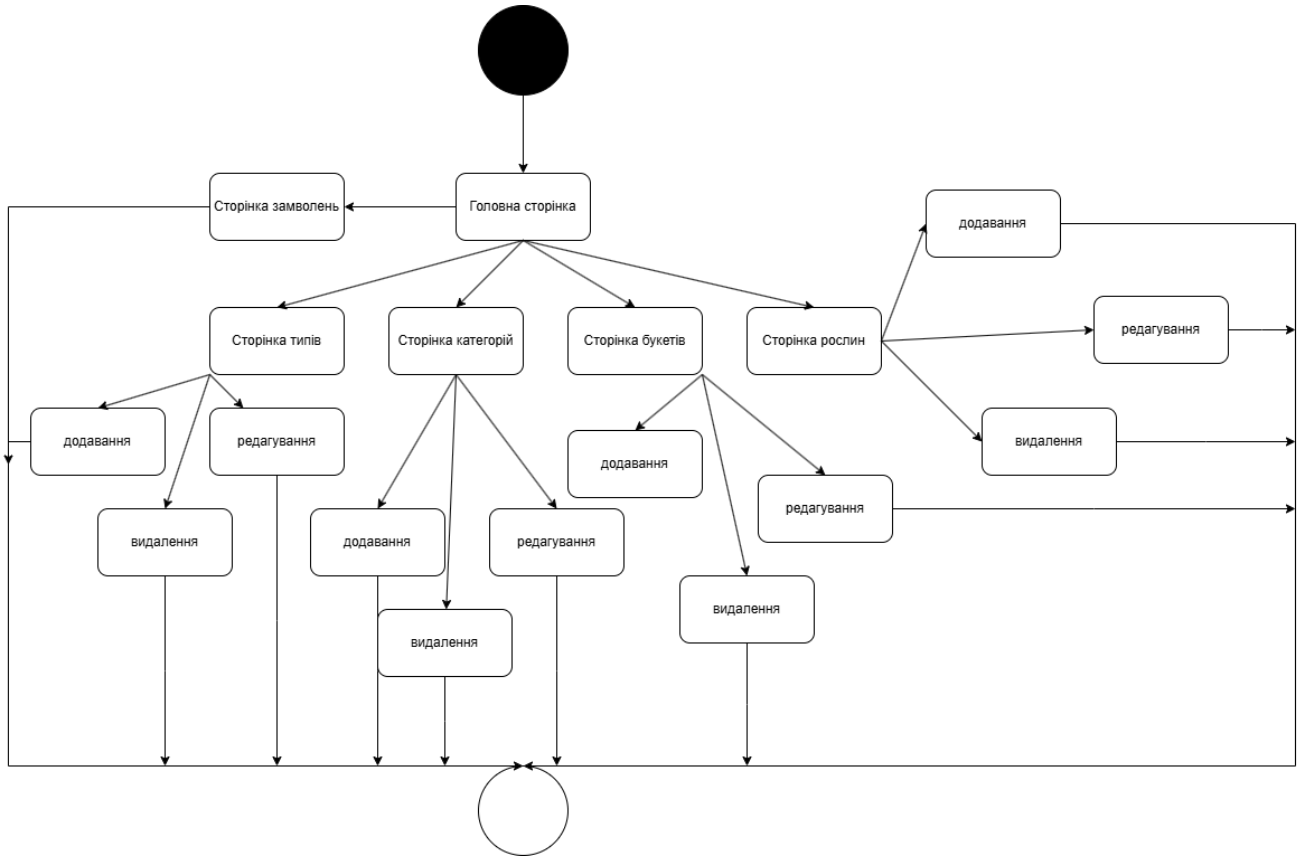


Рис. 2.6 Діаграма активності

2.4 Реалізація функціоналу Інтернет-магазину квітів

Почнемо з фронтенд частини. За підключення до API відповідають сервіси, приклад яких можна побачити в додатку А. Вони звертаються за певним Роутом до API та дістають або передають потрібну інформацію. Для структурування посилань використовуються певні енд-поінти, описані в Додатку В. Кожен сервіс має одну чи більше функцій, які відправляють або отримують дані за допомогою HTTP-запитів (POST, GET). Розглянемо детальніше один з сервісів, а саме – те,

що відповідає за функціонал LiqPay. В ньому міститься метод, який повертає платіжну форму та метод, який перевіряє статус оплати певного замовлення.

За діставання потрібної інформації з API відповідають сервіси, наприклад `paymentService` з методом `getPaymentForm`, який надсилає GET-запит на сервер для отримання HTML-форми оплати. Метод приймає об'єкт `orderData`, з якого формує параметри запиту (ID замовлення, сума, валюта, опис). За допомогою `fetch` надсилається запит на адресу `API_ENDPOINTS.PAYMENT_FORM` з цими параметрами. Якщо відповідь успішна, повертається текст форми (ймовірно, HTML). Якщо запит не вдається, виводиться помилка в консоль і вона повторно передається далі.

У цьому коді створено сервіс `paymentService` з методом `getPaymentForm`, який надсилає GET-запит на сервер для отримання HTML-форми оплати. Метод приймає об'єкт `orderData`, з якого формує параметри запиту (ID замовлення, сума, валюта, опис). За допомогою `fetch` надсилається запит на адресу `API_ENDPOINTS.PAYMENT_FORM` з цими параметрами. Якщо відповідь успішна, повертається текст форми (ймовірно, HTML). Якщо запит не вдається, виводиться помилка в консоль і вона повторно передається далі.

Далі перейдемо до реалізації кошику - Додаток С. Цей код створює глобальний контекст `CartContext` у React-додатку для керування кошиком покупок. Використовується хук `useState` для збереження товарів у кошику (`cart`), загальної суми (`total`) та HTML-форми оплати (`paymentForm`). Під час першого завантаження компонента `CartProvider` перевіряє `localStorage` і відновлює попередній стан кошика, якщо він збережений. При кожній зміні кошика оновлюється `localStorage`, обчислюється нова загальна сума й, якщо кошик не порожній, надсилається запит до `paymentService` для отримання HTML-форми оплати. Якщо кошик порожній — форма оплати скидається. Також реалізовано функції: додавання товару до кошика (з перевіркою наявності), видалення товару, оновлення кількості та повне очищення кошика. Компонент `CartContext.Provider` обгортає дочірні компоненти і надає їм доступ до всіх функцій і станів. Для зручності створено хук `useCart`, який дозволяє легко

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

працювати з кошиком у будь-якому компоненті, що знаходиться всередині CartProvider. За сам кошик відповідає файл у додатку D.

Компонент CartPage - сторінка, яка використовує дані з контексту CartContext через хук useCart. Основна його задача — відображати товари в кошику, дозволяти змінювати їхню кількість, видаляти товари та показувати форму оплати, якщо вона доступна. Тут визначається функція яка якщо кошик порожній (`cart.length === 0`), рендериться повідомлення про порожній кошик із кнопкою повернення до магазину (через Link).

Решта сторінок у додатку реалізована схожим чином, за всю роботу з API відповідають відповідні сервіси, які дістають дані, наприклад: у нас є сторінка товарів, в яку мають підзавантажитись квіти чи букети, ми звертаємось до сервісу букетів або квітів який робить запит за юрлом який вказаний в відповідному файлі та повертає нам квіти чи букети. Таким самим чином працюють пост і гет методи, якщо нам треба відправити дані на сторону API, ми це також робимо через сервіс.

Перейдемо до розгляду бекенд-частини сайту. Вона реалізована з використанням шаблону проєктування MVC (Model–View–Controller), що забезпечує чітке розділення логіки програми, інтерфейсу користувача та обробки даних. Такий підхід сприяє зручності у супроводі проєкту, дозволяє легко розширювати функціональність у майбутньому та спрощує внесення змін без порушення структури всієї системи.

База структури це ядро (Додаток E). Воно відповідає за запуск сесії, обробку маршрутів, виклик відповідного контролера та методу, обробку винятків і рендеринг сторінки.

Клас Controller (Додаток F) — це базовий клас для всіх контролерів у додатку. Він забезпечує універсальний механізм відображення представлень (views) через шаблони. Controller являє собою посередника між невидимою частиною сайту, тобто моделями (через які ми можемо взаємодіяти з БД за допомогою DbContext, лістинг якого можна побачити у додатку) та візуальною

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

частиною сайту, помітною для користувача (в цьому випадку – адміна, який буде слідкувати за дашбордом), за це і відповідають ці самі представлення (views).

Перейдемо до реалізації моделей програми (Додаток G). Модель відповідає за керування даними, логікою програми та правилами бізнес-логіки. Вона оперує даними, в цьому випадку – в базі даних. За саме діставання контенту з бази даних відповідають кілька файлів, це DbContext (додаток H), який відповідає саме за з'єднання з базою даних за допомогою PDO. За побудову самих звернень до БД відповідає інший допоміжний клас QueryBuilder (Додаток I), який містить всю основну логіку щодо побудови звернень. Клас QueryBuilder дозволяє безпосередньо формувати SELECT, INSERT, UPDATE та DELETE запити, а також додавати до них WHERE та ORDER BY частини, використовуючи методи select(), insert(), update(), delete(), where() і orderBy(). Всі параметри безпечно підставляються через масив \$statementParams для захисту від SQL-ін'єкцій. Метод execute() об'єднує сформовані частини запиту (\$primaryPart, \$wherePart, \$orderByPart), виконує підготовлений запит, і повертає результат: якщо це SELECT, то масив результатів, інакше — булевий результат виконання.

Усі дані, необхідні для підключення до зовнішніх сервісів, а також параметри взаємодії з API, зберігаються у спеціально створених конфігураційних файлах, що дозволяє централізовано керувати налаштуваннями системи та спрощує її підтримку. Детальна інформація наведена в Додатку J.

Висновки до другого розділу

В процесі проектування сайту (Інтернет-магазину квітів) були проаналізовані усі варіанти використання програмного продукту відповідно до вимог програмного забезпечення. На основі отриманого результату була спроектована структура проекту. Проектування включало в себе моделювання структури проекту та взаємодії між їх складовими частинами.

Також було спроектовано базу даних для зберігання даних сайту, таких як букетів, квітів, власних букетів, замовлень, магазинів та інших.

Було описано алгоритм роботи з платформою та описані послідовні пункти варіантів її використання.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						34

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ-МАГАЗИНОМ КВІТІВ

3.1. Порядок встановлення та налаштування параметрів платформи

Встановлення та налаштування Інтернет-магазину вимагає розгортання всіх його компонентів та налаштування як кожного елементу окремо, так і загальної системи в цілому. Для опису архітектури платформи було створено діаграму розгортання (рис. 3.1).

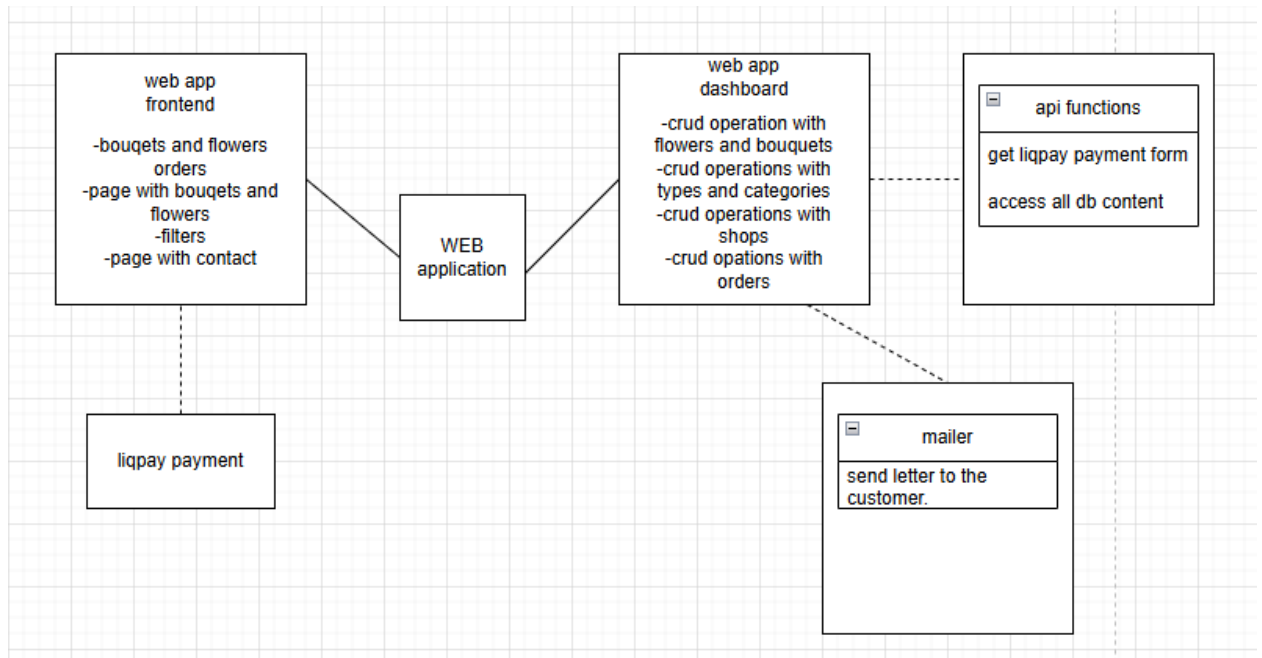


Рис. 3.1. Діаграма розгортання Інтернет-магазину квітів

Опис процесу розгортання.

Сайт можна розгорнути як на одному так і на двох серверах, структура сайту складається з бек частини та фронт частини.

Бек частина написана на PHP з використання патерну MVC, фронт частина написана на React.js.

У разі розгортання на одному сервері, спочатку налаштовується вебсервер (наприклад, Apache або Nginx), який обробляє HTTP-запити. Для підтримки PHP необхідно встановити відповідне середовище виконання (PHP версії 7.4 або вище), а також систему управління базами даних, таку як MySQL. Файли бекенд-частини розміщуються в кореневій директорії вебсервера. Створюється база даних, до якої підключається PHP-додаток, а також вносяться необхідні зміни у

конфігураційні файли для встановлення з'єднання з базою. Після цього перевіряється коректність роботи маршрутизації згідно з принципами MVC: запити від користувача мають передаватися контролерам, які у свою чергу взаємодіють з моделями та оновлюють уявлення (View).

Для фронтенду встановлюється Node.js та менеджер пакетів npm. Після завантаження вихідного коду React-застосунку необхідно виконати команду npm install, яка завантажує всі залежності проєкту. Далі за допомогою npm run build створюється оптимізована збірка, яка розміщується в публічній директорії вебсервера або в окремій директорії на сервері, якщо використовується окреме розгортання. У цьому випадку вебсервер налаштовується на обслуговування статичних файлів, зокрема HTML, CSS, JS. Якщо клієнтська частина розгортається на окремому сервері або домені, слід забезпечити правильне налаштування запитів до API, тобто вказати актуальні URL бекенд-сервера в налаштуваннях React-застосунку.

У випадку використання двох серверів — один виконує роль API-сервера (бекенд), а інший — сервера клієнтського інтерфейсу (фронтенд). У такому варіанті важливо забезпечити CORS (Cross-Origin Resource Sharing), щоб клієнтська частина мала доступ до ресурсів API. Також варто налаштувати доменне ім'я або піддомени для обох частин системи, організувавши сертифікацію HTTPS для безпечної передачі даних.

Завершальним етапом є тестування усіх основних функцій: доступність клієнтської частини, коректність взаємодії з API, підключення до бази даних, а також працездатність платіжного модуля LiqPay. Усе це гарантує, що сайт працює стабільно та готовий до використання кінцевими користувачами.

3.2. Структура інтерфейсу Інтернет-магазину квітів

На головній сторінці платформи знаходяться яскраві фото, що описують тематику магазину та дещо збільшуються при наведенні та слоган. Головне меню з посиланнями на сторінки «Букети», «Рослини», «Контакти» та «Кошик» знаходиться вище, а знизу дві доступні категорії товарів «Букети» та «Рослини» з кнопками переходу на відповідні сторінки.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

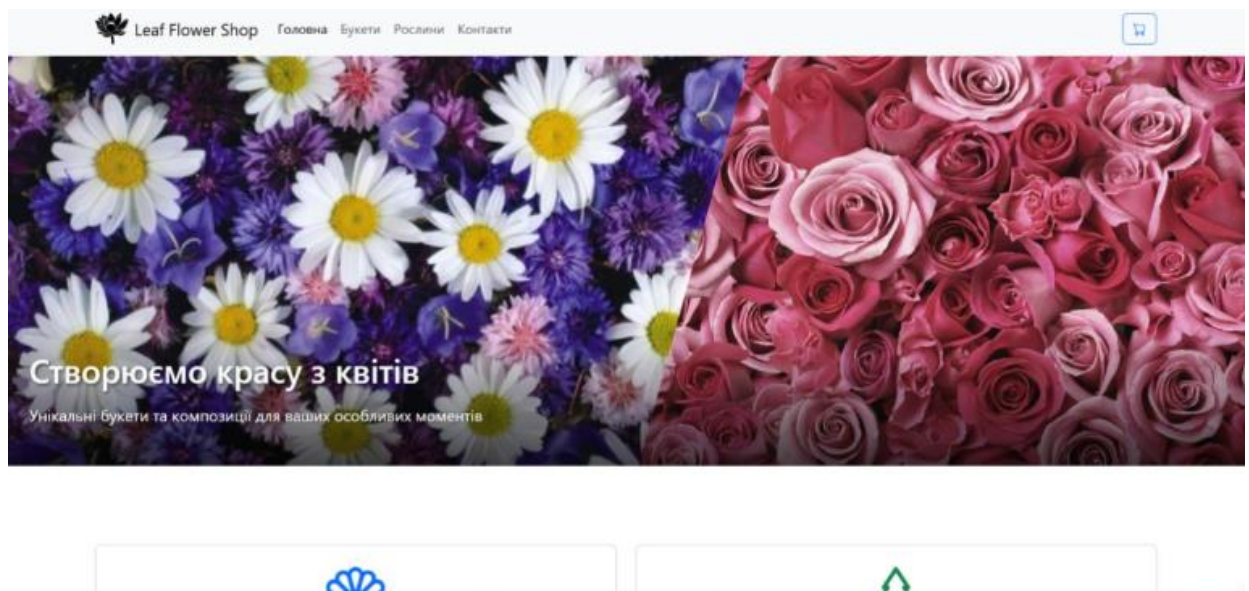


Рис. 3.2. Головна сторінка

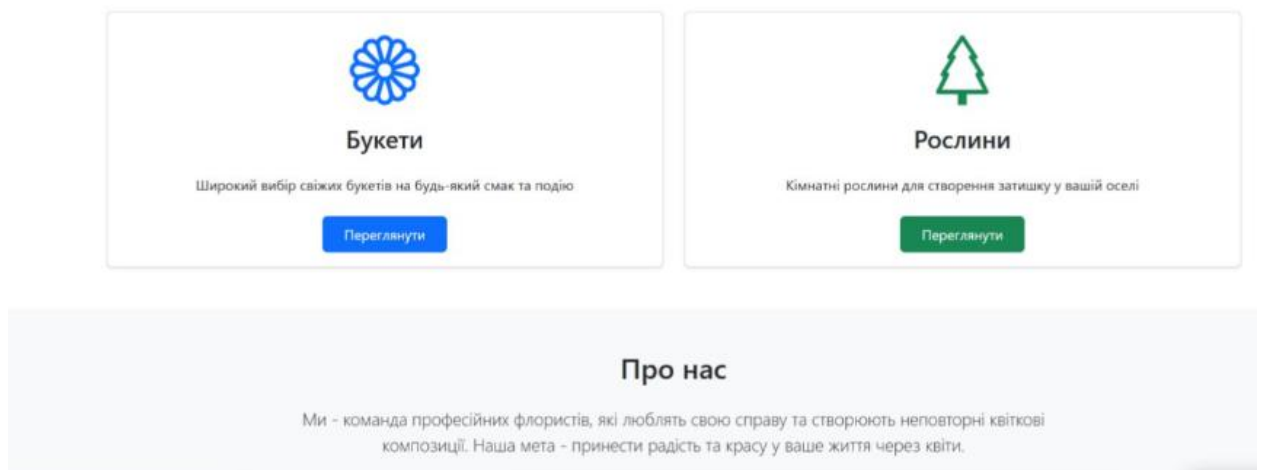


Рис. 3.3. Продовження головної сторінки

На сторінці «Букети» відображається каталог усіх наявних букетів. Кожна позиція товару містить такі елементи: фото, назва, ціна, квіти, що входять в букет та їх кількість, та кнопка «Додати до кошика». З лівого боку від каталогу розташований блок «Фільтр» з такими можливостями: пошук за назвою, пошук за ціновим діапазоном та пошук за квітами, що входять в букет.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

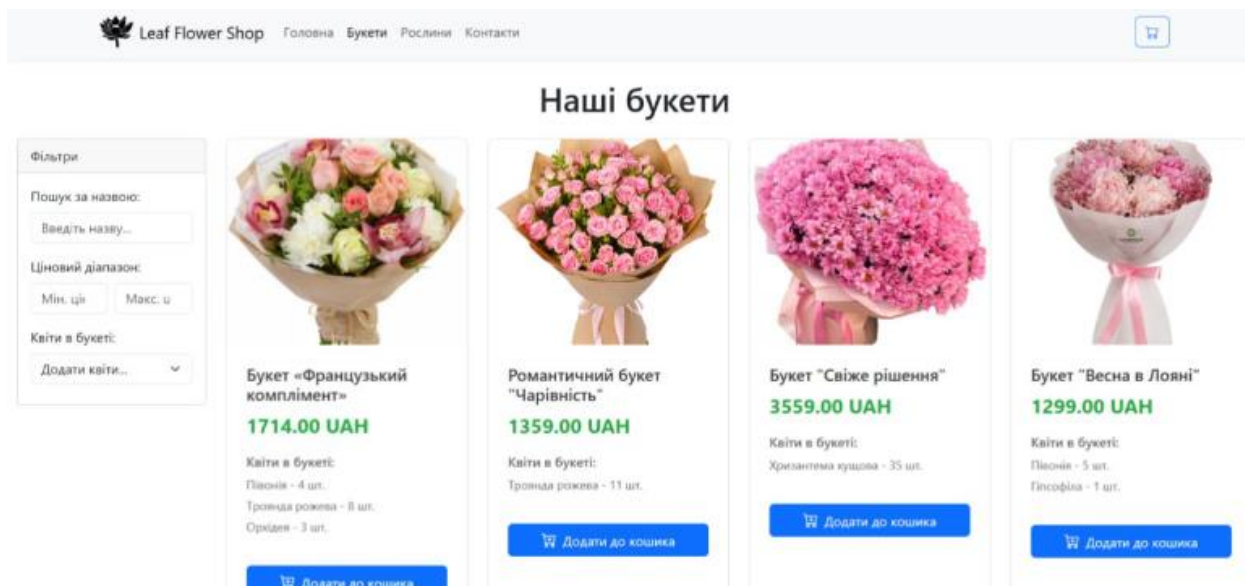


Рис. 3.4. Сторінка «Букети»

Приклад роботи фільтра показано нижче.

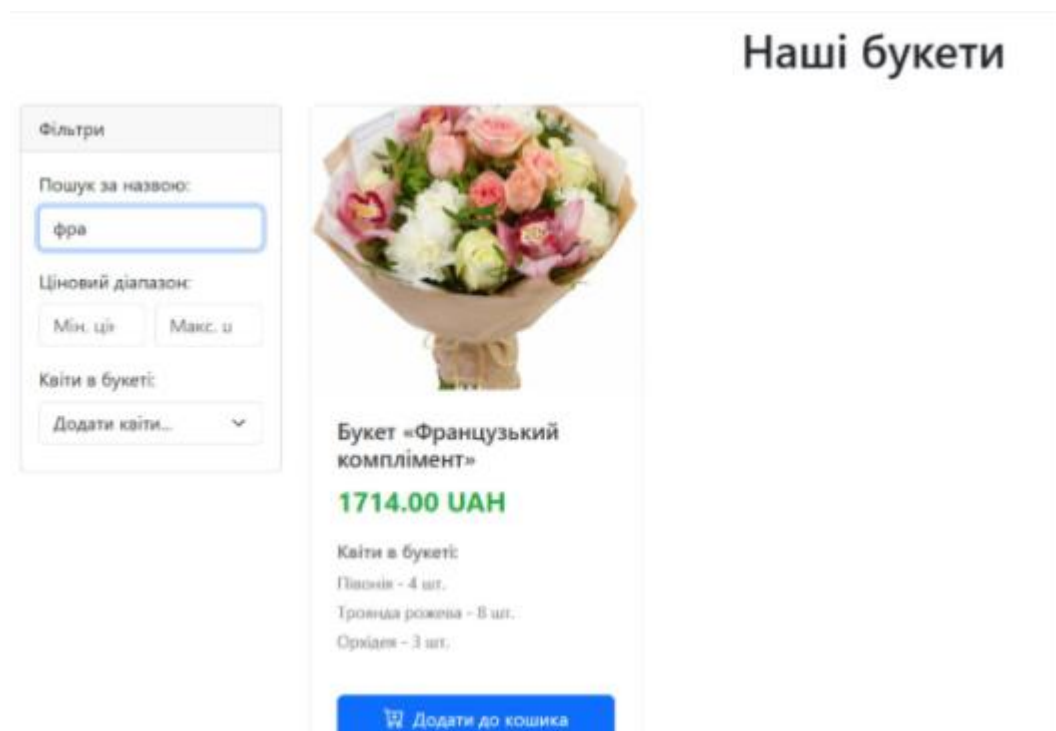


Рис. 3.5. Пошук букета за назвою

Сторінка «Рослини» дуже схожа на сторінку з букетами, адже позиції товарів містять такі елементи: фото, назва, ціна, висота рослини та кнопка додавання в кошик, проте блок «Фільтр» має відмінності: присутній пошук за назвою та цінова категорія, а також можна обрати категорію рослини (наприклад, квіткові чи садові), термін життя (наприклад, багаторічні чи однорічні), а також чи наявна рослина в букетах.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

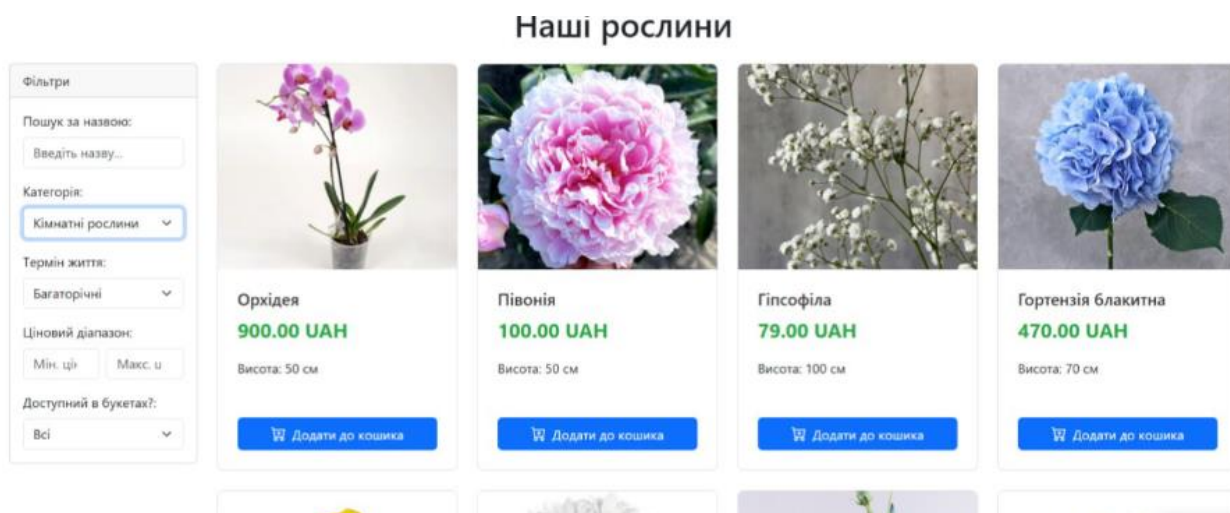


Рис. 3.6. Сторінка «Рослини»

Класична іконка Кошику в хедері відображає додану до нього кількість товару. Сторінка «Кошик» виглядає наступним чином: відображаються обрані товари, їх кількість можна змінювати, або видалити позицію з кошика, залежно від кількості товару його вартість змінюється. Також на кожній позиції на синьому фоні відображається категорія товару, натиснувши на яку можна перейти на відповідну сторінку. Праворуч знаходиться форма оформлення замовлення. На ній відображена кількість товарів, загальна вартість, далі необхідно заповнити поля з контактною інформацією та обрати спосіб отримання замовлення.

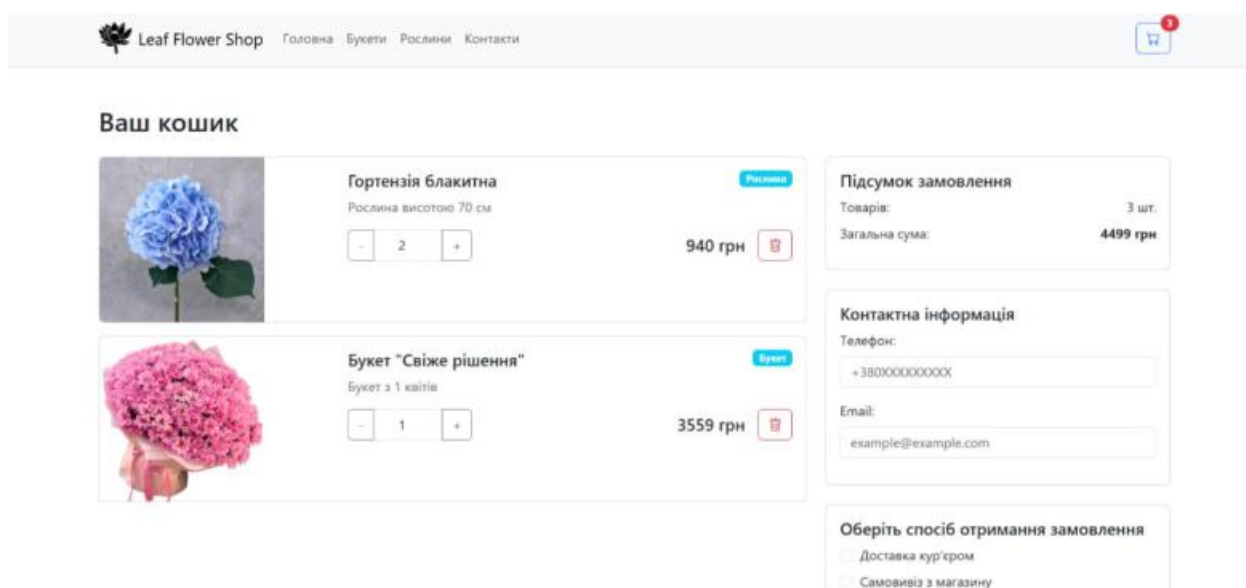


Рис. 3.7. Сторінка «Кошик»

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

Залежно від обраного способу отримання зовлення на екрані з'являється додаткова форма (якщо це самовивіз, то необхідно обрати магазин, а якщо доставка кур'єром, то вказати адресу).

Контактна інформація

Телефон:

+380674187940

Email:

ksuhatop228@gmail.com

Оберіть спосіб отримання замовлення

☐ Доставка кур'єром

☒ Самовивіз з магазину

Оберіть магазин:

☐ Хлібна 14

☒ Київська 12

☐ Нардицька 41

☐ Князів Острозьких 18

Оплата замовлення

Сплатити

Рис. 3.8. Приклад оформлення замовлення

Після натиснення на кнопку «Сплатити» з'являється сторінка оплати

Тестовий режим

QR-код для оплати

Використовуйте Privat24

LIQPAY

Дані про оплату

Замовлення товарів (2 шт.)

До сплати: 4499.00 UAH

24 Pay

Оплатити через G Pay

або

Privat24 24 Privat24

Рис. 3.9. Сторінка оплати

Тим часом на сторінці «Кошик» з'являється повідомлення про перевірку статусу оплати

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

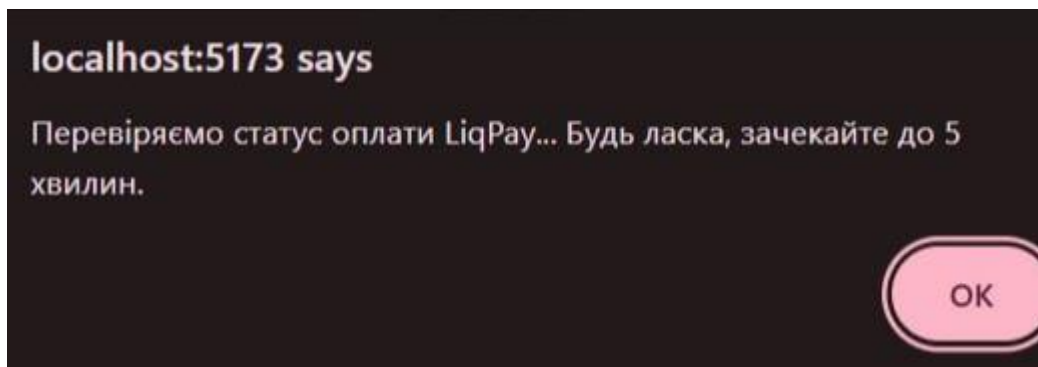


Рис. 3.10. Перевірка статусу оплати

Після того, як платіж підтверджено, повідомлення вже говорить про успішність оплати та виводить номер замовлення

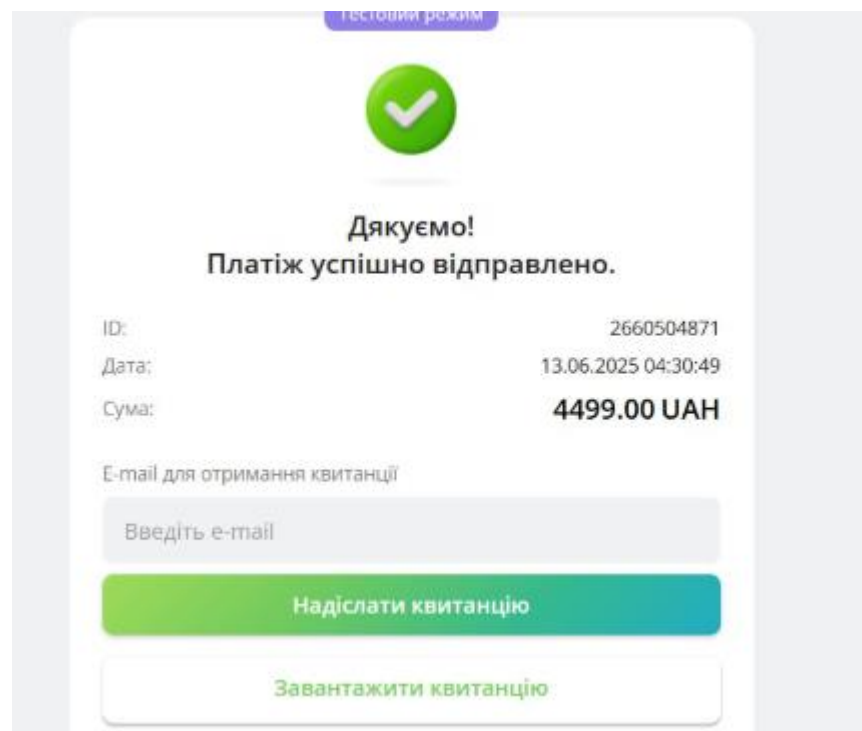


Рис. 3.11. Оплата успішна

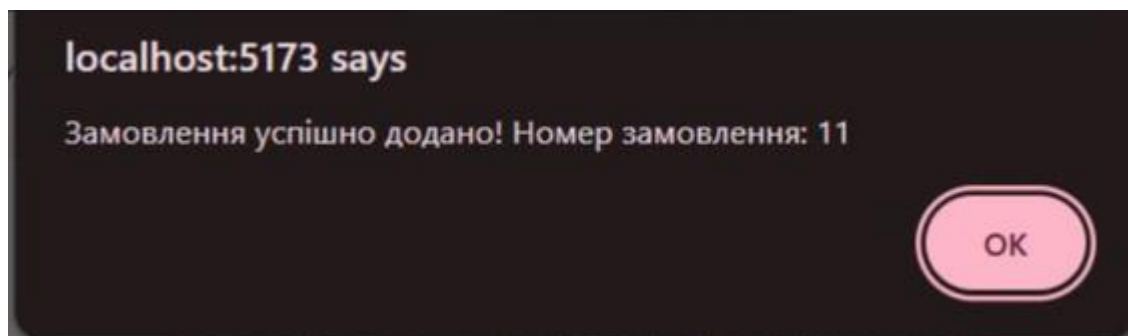


Рис. 3.12. Оплата успішна

Одразу після завершення платежу на електронну адресу, зазначену у формі оформлення замовлення, надійде повідомлення про замовлення, що містить його

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

номер, дату, вартість, спосіб доставки та інформацію по сам товар.

Дякуємо за ваше замовлення №10

Дата: 2025-06-13

Сума: 1359 грн

Самовивіз з магазину ID: 3

Ваше замовлення:

- 🌹 Букет: Романтичний букет "Чарівність" — 1 шт. — 1359₴
 - Троянда рожева — 11 шт.

Очікуйте дзвінка нашого менеджера для підтвердження замовлення.

Рис. 3.13. Повідомлення на електронній пошті

Тепер переходимо до адмін частини сайту. Одразу після оформлення клієнтом замовлення, воно з'являється на адмін сторінці «Замовлення» з датою замовлення, контактною інформацією, способом доставки, вартістю та двома кнопками (Перегляд та Видалити).

Квіти РОСЛИНИ БУКЕТИ ЗАМОВЛЕННЯ

Управління

Список замовлень

ID	Дата	Телефон	Email	Доставка	Ціна	Дії
5	2025-06-12	+380955555555	qqqq@gmail.com	Ні	1714 грн	<button>Перегляд</button> <button>Видалити</button>
6	2025-06-12	+380943858345	debis@qq.com	Так	4858 грн	<button>Перегляд</button> <button>Видалити</button>
7	2025-06-13	+380674187940	ksuhatop228@gmail.com	Ні	100 грн	<button>Перегляд</button> <button>Видалити</button>
8	2025-06-13	+380674187940	ipz215_pov@student.ztu.edu.ua	Так	3702 грн	<button>Перегляд</button> <button>Видалити</button>
9	2025-06-13	+380977848583	ipz215_pov@student.ztu.edu.ua	Ні	200 грн	<button>Перегляд</button> <button>Видалити</button>
10	2025-06-13	+380674187940	ipz215_pov@student.ztu.edu.ua	Ні	1359 грн	<button>Перегляд</button> <button>Видалити</button>

Рис. 3.14. Сторінка «Замовлення»

Натиснемо на кнопку «Перегляд» останнього замовлення. Сторінка містить дату замовлення, контактну інформацію, спосіб доставки (адреса, якщо обрано доставку кур'єром), вартість та саме замовлення

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

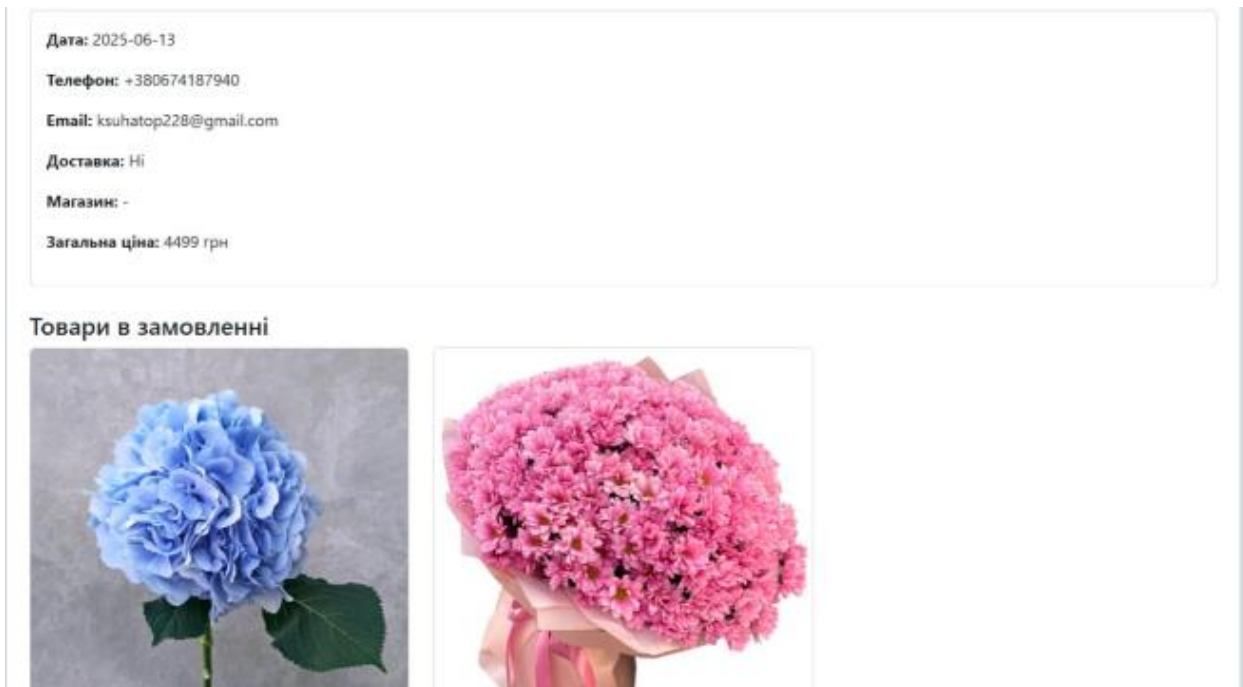


Рис. 3.15. Сторінка окремого замовлення

Головна сторінка Адміна містить 4 кнопки «Рослини», «Букети», «Категорії», «Типи рослин» для подальших дій з ними.



Рис. 3.16. Головна сторінка адмін панелі

Відкриємо сторінку з Рослинами. На ній відображається кнопка «Додати рослину», каталог рослин та відповідні кнопки Редагування та Видалення до кожної з них.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

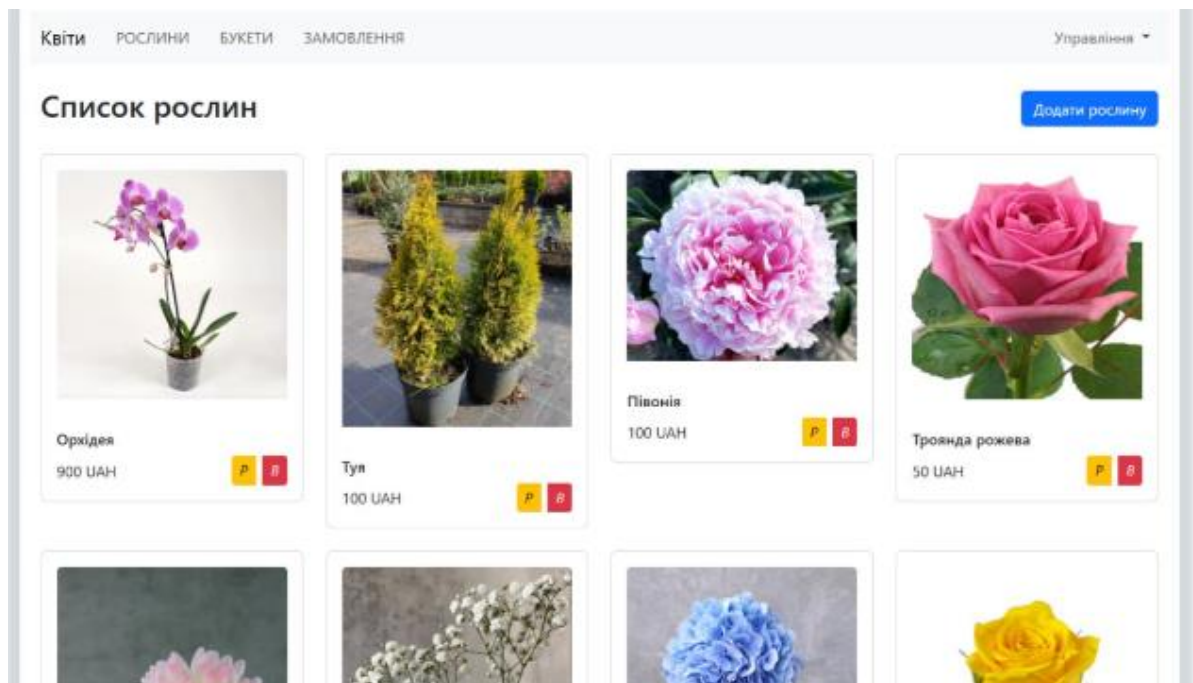


Рис. 3.17. Сторінка «Рослини»

Сторінка Додавання нової рослини вміщає в себе форму, де необхідно зазначити категорію і тип рослини, назву, ціну, висоту, додати зображення та зазначити чи може входити рослина до складу букету.

Рис. 3.18. Сторінка додавання нової рослини

Сторінка редагування рослини містить ту ж саму форму, але вже заповнену, яку можна відредагувати.

Сторінка «Букети» виглядає схожим чином.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

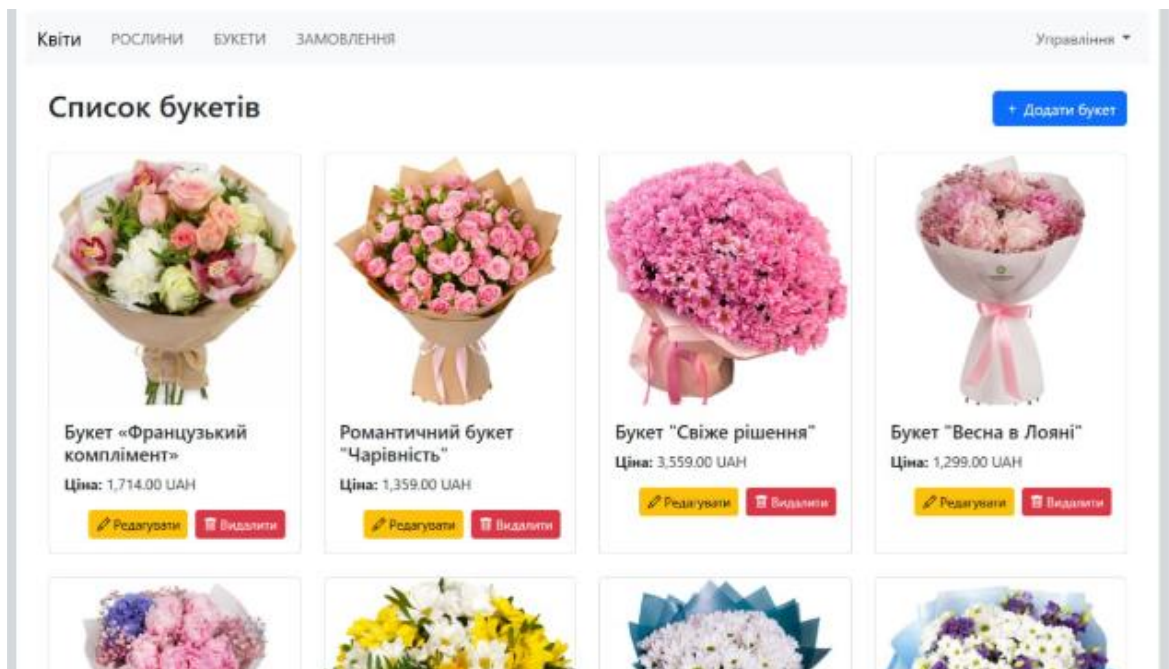


Рис. 3.19. Сторінка «Букети»

Сторінка додавання нового букету має форму з такими полями: Назва букета, іна букета, Зображення, вибір Квіток (одного виду чи декількох) у букеті та їх кількість.

Рис. 3.20. Сторінка «Додати букет»

Сторінка редагування букету містить ту ж саму форму, але вже заповнену, яку можна відредагувати.

Рис. 3.21. Сторінка «Редагувати букет»

Крім того, на сторінці Адміна можна редагувати категорії та типи рослин та додавати нові (посилання на відповідні сторінки знаходяться на головній сторінці).

ID	Назва	Дії
1	Кімнатні рослини	P B
2	Садові рослини	P B

Рис. 3.22. Сторінка «Категорії рослин»

Рис. 3.23. Сторінка «Редагувати категорію»

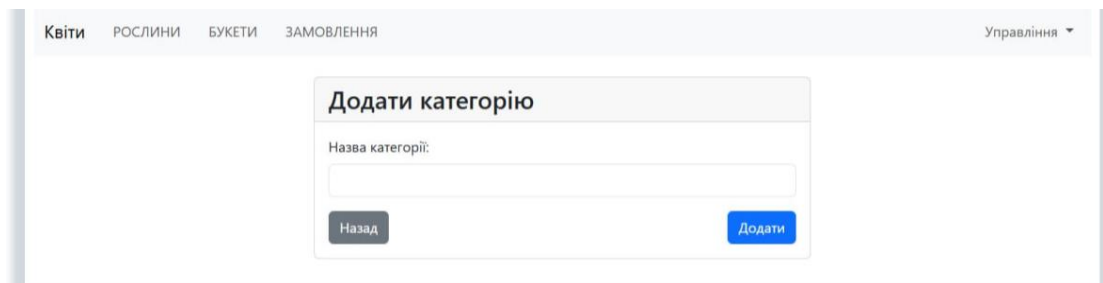
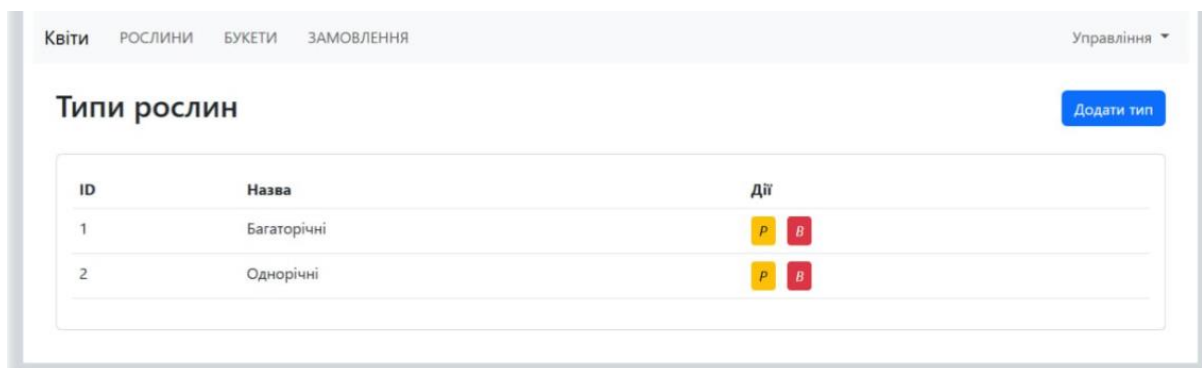


Рис. 3.24. Сторінка «Додати категорію»

Такі самі дії можна зробити з типами рослин.



ID	Назва	Дії
1	Багаторічні	Р В
2	Однорічні	Р В

Рис. 3.25. Сторінка «Типи рослин»

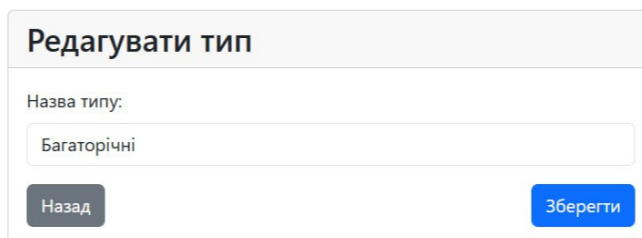


Рис. 3.26. Сторінка «Редагувати тип»



Рис. 3.27. Сторінка «Додати тип»

Попередження про видалення типу рослин виглядатиме як спливаюче повідомлення.

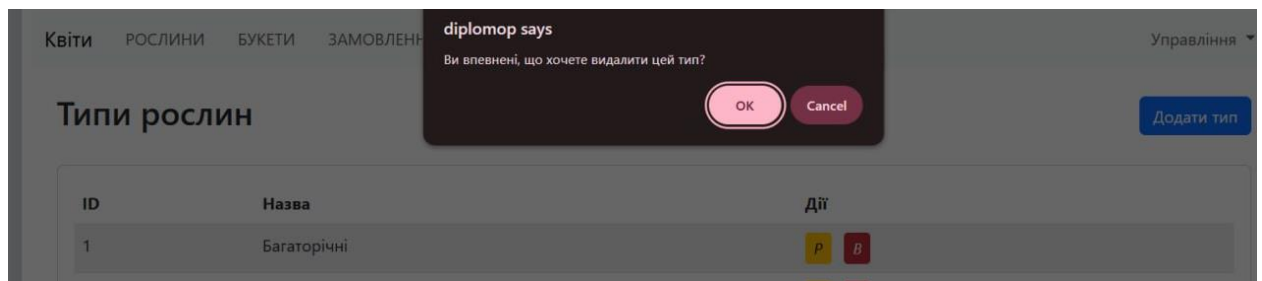


Рис. 3.28. Попередження про видалення типу рослин

Також можна змінювати інформацію про наявні магазини або додавати нові

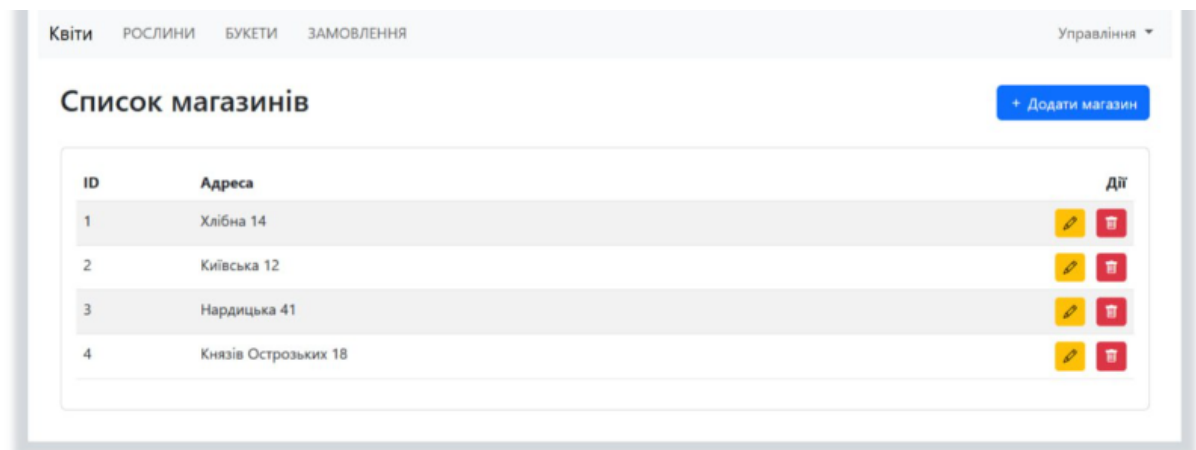


Рис. 3.29. Сторінка «Список магазинів»

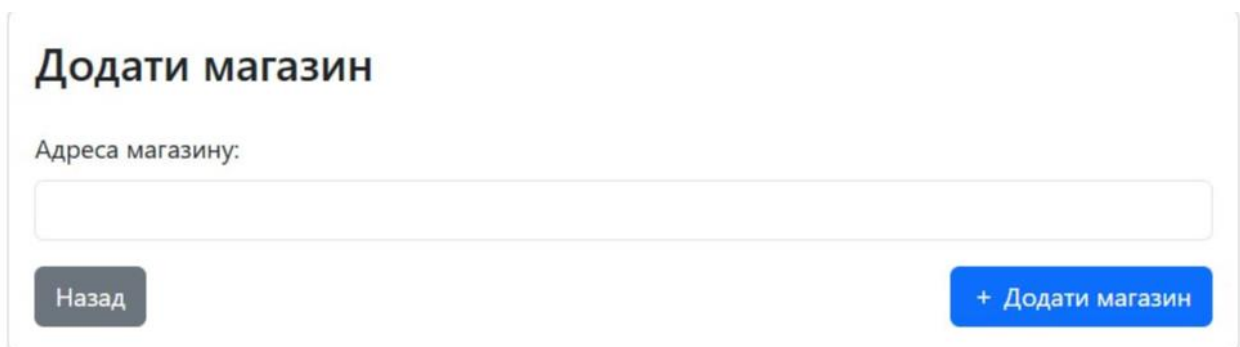


Рис. 3.30. Сторінка «Додати магазин»

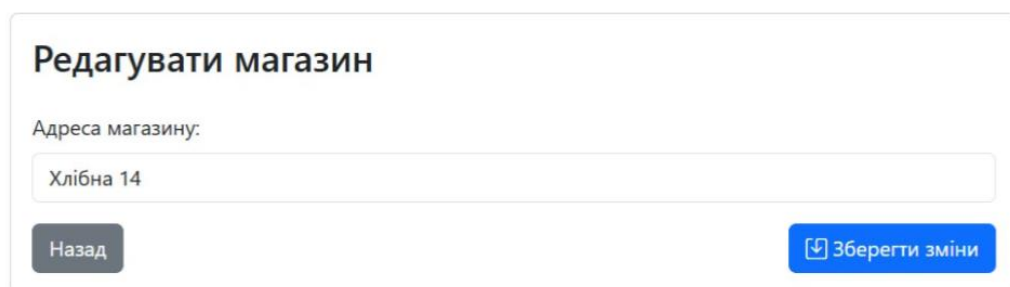


Рис. 3.31. Сторінка «Редагувати магазин»

Стрінкою, що носить додаткову інформаційну інформацію для користувачів є «Контакти», що повідомляє про місце розташування, контактний номер, електронну адресу, графік та прикріплену гугл-мапу для зручнішого пошуку магазину офлайн.

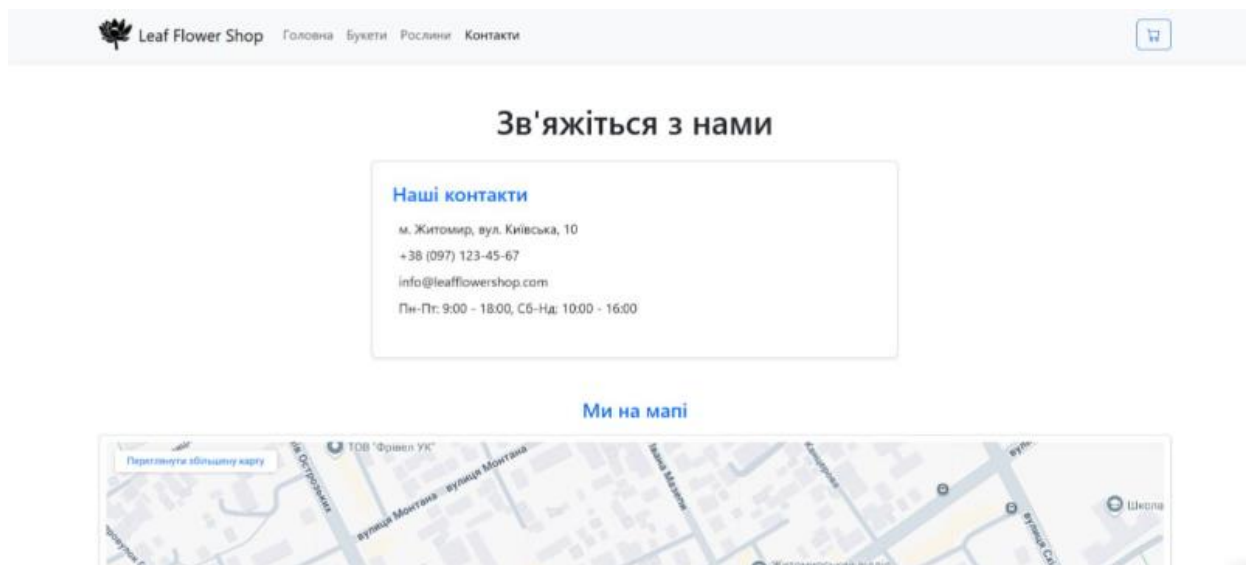


Рис. 3.32. Сторінка «Контакти»

3.3. Тестування програмного продукту

Тестування Інтернет-магазину квітів є критичним етапом усього процесу розробки, оскільки саме на цьому етапі перевіряється коректність роботи всіх функціональних модулів, надійність взаємодії клієнтської та серверної частин, а також загальна зручність користування інтерфейсом. Основною метою тестування є виявлення та усунення помилок ще до моменту запуску системи в реальне використання, що забезпечує стабільну роботу сайту для кінцевих користувачів.

У процесі тестування було проведено перевірку функціональності усіх основних можливостей системи. З боку клієнта це включає навігацію сторінками, перегляд каталогу букетів і рослин, формування власних композицій, додавання товарів до кошика, редагування вмісту кошика, оформлення замовлення, а також успішну інтеграцію з платіжною системою LiqPay. Кожна з цих дій проходила перевірку на предмет правильності логіки виконання, відсутності збоїв та помилок, а також відповідності очікуваному результату.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						49

З боку адміністратора тестування охоплювало всі CRUD-операції для таких сутностей, як магазини, типи, категорії, рослини та букети. Окрім цього, була перевірена функціональність перегляду замовлень і правильність відображення змін у системі. Особливу увагу було приділено безпеці введення даних та перевірці на обробку некоректних або шкідливих запитів.

Графічний інтерфейс також був протестований на відповідність базовим вимогам юзабіліті. Інтерфейс мав бути інтуїтивно зрозумілим, доступним для сприйняття, а всі елементи управління — зручно розташованими та функціонально активними. Час відгуку системи при взаємодії з користувачем не перевищував трьох секунд, що відповідає сучасним стандартам продуктивності вебзастосунків.

За результатами тестування було виявлено та усунуто кілька незначних помилок у логіці взаємодії між компонентами системи, що дозволило досягти стабільної та надійної роботи платформи в умовах реального користування.

Висновки до третього розділу

У цьому розділі було детально розглянуто процес розгортання платформи, включаючи опис її архітектури та взаємодії між основними складовими системи. Кожен компонент отримав пояснення щодо свого призначення та зв'язку з іншими елементами платформи, що дало змогу краще зрозуміти логіку побудови та організації системи.

Також було описано інтерфейс створеного програмного продукту. Огляд охопив усі функціональні можливості, які надаються користувачам залежно від їхньої ролі в системі.

Проведене тестування, що підтвердило коректну роботу всіх основних функціональних модулів Інтернет-магазину квітів. Згідно з результатами перевірок, як серверна, так і клієнтська частини платформи стабільно виконують свої завдання, інтерфейс залишається зручним та інтуїтивно зрозумілим для користувача. Виявлені незначні недоліки були оперативно усунуті, що забезпечило підвищення загальної надійності та якості програмного продукту.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						50

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано платформу Інтернет-магазину квітів. Проведено детальний аналіз предметної області та досліджено конкурентні рішення — популярні сервіси з онлайн-продажу квітів. На основі цього аналізу були визначені ключові функціональні та нефункціональні вимоги до майбутнього програмного продукту.

У процесі розробки було створено архітектуру системи, обрано відповідні технології та визначено принципи взаємодії між її компонентами. Платформа побудована за архітектурним патерном MVC, що дозволило ефективно розділити логіку роботи з даними (модель), управління діями користувача (контролер) та інтерфейс (представлення). Окрему увагу приділено проектуванню структури бази даних для зберігання інформації про товари, замовлення, магазини, категорії, типи рослин та користувачів.

Також описано алгоритм взаємодії користувача з платформою — від ознайомлення з асортиментом, додавання товарів у кошик і до оформлення замовлення з оплатою через сервіс LiqPay. Деталізовано функціональні можливості додатку як для звичайного користувача, так і для адміністратора, включаючи додавання, редагування та видалення елементів товарної структури.

У роботі представлено схему розгортання Інтернет-магазину, описано компоненти системи, їх взаємозв'язок та інтерфейс користувачів. Також продемонстровано, як реалізується візуалізація інформації в адміністративній панелі.

Розроблений Інтернет-магазин забезпечує прозору та зручну взаємодію між користувачами та системою. Інтуїтивний інтерфейс дає змогу швидко орієнтуватися в асортименті, а адміністратору — ефективно керувати контентом магазину. Платформа є масштабованою, продуктивною та готовою до впровадження в реальних умовах.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Flowers.ua – URL: <https://flowers.ua/>
2. UFL.ua – URL: <https://www.u-f-l.net/>
3. Flora Queen – URL: <https://www.floraqueen.com/ru/c/sending-flowers-all-flowers>
4. LiqPay – SDK – URL: <https://www.liqpay.ua/en/doc/sdk?utm>
5. React with PHP: A Comprehensive Guide – URL: <https://www.esparkinfo.com/softwaredevelopment/technologies/reactjs/react-with-php?utm>
6. Building a Scalable React App – URL: <https://welovedevs.com/articles/scalable-react-app/?utm>
7. PHP and MVC – URL: <https://wpwebinfotech.com/blog/php-and-mvc/?utm>
8. PHPMailer – URL: <https://github.com/PHPMailer/PHPMailer>
9. Martin R. Clean Code: A Handbook of Agile Software Craftsmanship / Robert C. Martin. — Boston: Prentice Hall, 2008. — 464 с.
10. PHP – URL: <https://www.php.net/>
11. React – URL: <https://react.dev/>
12. PHPMailer Guide – URL: <https://mailtrap.io/blog/phpmailer/>
13. Gmail пошта. Налаштування SMTP – URL: <https://ukrline.com.ua/ua/blog/101973.php>
14. MySQL – URL: <https://dev.mysql.com/doc/>
15. AXIOS for the browser and node.js – URL: <https://axios-http.com/>
16. React Router – URL: <https://reactrouter.com/>
17. Resources for Developers by Developers – URL: <https://developer.mozilla.org/en-US/>
18. PHP Standards Recommendations – URL: <https://www.php-fig.org/psr/>
19. DigitalOcean Community Tutorials – URL: <https://www.digitalocean.com/community/tutorials>
20. OWASP Top Ten – URL: <https://owasp.org/www-project-top-ten/>

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

21. Freeman, E., Robson, E. Head First Design Patterns. – 2nd ed. – O'Reilly Media, 2020. – 694 p.
22. Crockford, D. JavaScript: The Good Parts. – Sebastopol: O'Reilly Media, 2008. – 176 p.
23. Flanagan, D. JavaScript: The Definitive Guide. – 7th ed. – Sebastopol: O'Reilly Media, 2020. – 706 p.
24. Nystrom, R. Game Programming Patterns. – Genever Benning, 2014. – 354 p.
25. Gibson Bond, J. Introduction to Game Design, Prototyping, and Development. – Boston: Addison-Wesley, 2017. – 944 p.
26. Richter, J. CLR via C#. – 4th ed. – Redmond: Microsoft Press, 2012. – 826 p.
27. Shiffman, D. The Nature of Code. – Self-published, 2012. – 498 p.
28. Short, T. X., Adams, T. Procedural Generation in Game Design. – Boca Raton: CRC Press, 2017. – 288 p.
29. Rogers, S. Level Up! The Guide to Great Video Game Design. – 2nd ed. – Hoboken: Wiley, 2014. – 552 p.
30. Swink, S. Game Feel: A Game Designer's Guide to Virtual Sensation. – Boca Raton: CRC Press, 2008. – 376 p.
31. Adams, E., Rollings, A. Fundamentals of Game Design. – 3rd ed. – New York: New Riders, 2014. – 624 p.

		Павленко О. В.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О.В.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Додатки

```

const API_BASE_URL = 'http://diplomop';

export const API_ENDPOINTS = {
  BOUQUETS: `${API_BASE_URL}/api/getBouquet`,
  PLANTS: `${API_BASE_URL}/api/getPlant`,
  CATEGORIES: `${API_BASE_URL}/api/getCategories`,
  TYPES: `${API_BASE_URL}/api/getTypes`,
  PAYMENT_FORM: `${API_BASE_URL}/liqpayApi/getForm`,
  SHOPS: `${API_BASE_URL}/api/getShops`,
  ADD_ORDER: `${API_BASE_URL}/api/addOrder`,
  PAYMENT_STATUS: `${API_BASE_URL}/liqpayApi/checkOrderStatus`
};

```

```

import { API_ENDPOINTS } from '../config/api';

export const paymentService = {
  async getPaymentForm(orderData) {
    try {
      const queryParams = new URLSearchParams({
        order_id: orderData.order_id,
        amount: orderData.amount,
        currency: orderData.currency || 'UAH',
        description: orderData.description || ''
      });

      const response = await
fetch(`${API_ENDPOINTS.PAYMENT_FORM}?${queryParams}`, {
      method: 'GET',
      headers: {
        'Content-Type': 'application/json',
      }
    });

      if (!response.ok) {
        throw new Error('Failed to get payment form');
      }

      const data = await response.text();
      return data;
    } catch (error) {
      console.error('Error getting payment form:', error);
      throw error;
    }
  },

  async getPaymentStatus(orderId) {
    try {
      const queryParams = new URLSearchParams({

```

```

        order_id: orderId,
    });

    const response = await
    fetch(`${API_ENDPOINTS.PAYMENT_STATUS}?${queryParams}`, {
        method: 'GET',
        headers: {
            'Content-Type': 'application/json',
        }
    });

    if (!response.ok) {
        const errorBody = await response.json().catch(() =>
        response.text());
        throw new Error(`Failed to get payment status: ${response.status}
        ${response.statusText} - ${JSON.stringify(errorBody)}`);
    }

    const data = await response.json();
    return data;
} catch (error) {
    console.error(`Error getting payment status for order ID ${orderId}:`,
    error);
    throw error;
}
}
};

```

Додаток С

```

import React, { createContext, useContext, useState, useEffect } from 'react';
import { paymentService } from '../services/paymentService';

const CartContext = createContext();

export const CartProvider = ({ children }) => {
    const [cart, setCart] = useState([]);
    const [total, setTotal] = useState(0);
    const [paymentForm, setPaymentForm] = useState(null);
    const [liqPayOrderId, setLiqPayOrderId] = useState(null);
    const [isInitializing, setIsInitializing] = useState(true);

    const calculateTotal = (cartData = cart) => {
        const newTotal = cartData.reduce((sum, item) => sum + (item.price *
        item.quantity), 0);
        setTotal(newTotal);
        return newTotal;
    };

    const updatePaymentForm = async (currentTotal, currentCart = cart) => {
        try {
            const newOrderId = Date.now().toString();

```

```

    setLiqPayOrderId(newOrderId);

    const orderData = {
      order_id: newOrderId,
      amount: currentTotal,
      currency: 'UAH',
      description: `Замовлення товарів (${currentCart.length} шт.)`
    };
    const form = await paymentService.getPaymentForm(orderData);
    setPaymentForm(form);
  } catch (error) {
    console.error('Error getting payment form:', error);
    setPaymentForm(null);
    setLiqPayOrderId(null);
  }
};

useEffect(() => {
  const initializeCart = async () => {
    const savedCart = localStorage.getItem('cart');
    if (savedCart) {
      const parsedCart = JSON.parse(savedCart);
      setCart(parsedCart);
      const newTotal = parsedCart.reduce((sum, item) => sum + (item.price
* item.quantity), 0);
      setTotal(newTotal);
      if (parsedCart.length > 0) {
        await updatePaymentForm(newTotal, parsedCart);
      }
    }
    setIsInitializing(false);
  };
  initializeCart();
}, []);

useEffect(() => {
  if (isInitializing) return;

  localStorage.setItem('cart', JSON.stringify(cart));
  const currentTotal = calculateTotal(cart);
  if (cart.length > 0) {
    updatePaymentForm(currentTotal);
  } else {
    setPaymentForm(null);
    setLiqPayOrderId(null);
  }
}, [cart, isInitializing]);

const addToCart = (item) => {
  setCart(prevCart => {

```

```

    const existingItem = prevCart.find(cartItem => cartItem.id === item.id);
    if (existingItem) {
      return prevCart.map(cartItem =>
        cartItem.id === item.id
          ? { ...cartItem, quantity: cartItem.quantity + 1 }
          : cartItem
      );
    }
    return [...prevCart, { ...item, quantity: 1 }];
  });
};

const removeFromCart = (itemId) => {
  setCart(prevCart => prevCart.filter(item => item.id !== itemId));
};

const updateQuantity = (itemId, quantity) => {
  setCart(prevCart =>
    prevCart.map(item =>
      item.id === itemId ? { ...item, quantity } : item
    )
  );
};

const clearCart = () => {
  setCart([]);
  setTotal(0);
  setPaymentForm(null);
  setLiqPayOrderId(null);
};

return (
  <CartContext.Provider value={{
    cart,
    total,
    paymentForm,
    liqPayOrderId,
    addToCart,
    removeFromCart,
    updateQuantity,
    clearCart
  }}>
    {children}
  </CartContext.Provider>
);
};

export const useCart = () => {
  const context = useContext(CartContext);
  if (!context) {

```

```

        throw new Error('useCart must be used within a CartProvider');
    }
    return context;
};

```

Додаток D

```

import React, { useState, useEffect } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { useCart } from '../contexts/CartContext';
import { getShops } from '../services/shopService';
import { addOrder } from '../services/orderService';
import { paymentService } from '../services/paymentService';

const CartPage = () => {
    const { cart, removeFromCart, updateQuantity, total, paymentForm, clearCart,
    liqPayOrderId } = useCart();
    const navigate = useNavigate();

    const [deliveryOption, setDeliveryOption] = useState('');
    const [deliveryAddress, setDeliveryAddress] = useState('');
    const [shops, setShops] = useState([]);
    const [loadingShops, setLoadingShops] = useState(false);
    const [shopError, setShopError] = useState(null);
    const [customerPhone, setCustomerPhone] = useState('');
    const [customerEmail, setCustomerEmail] = useState('');
    const [deliveryDate, setDeliveryDate] = useState('');
    const [deliveryTime, setDeliveryTime] = useState('');
    const [selectedShopId, setSelectedShopId] = useState(null);

    useEffect(() => {
        if (deliveryOption === 'pickup' && shops.length === 0) {
            const fetchShops = async () => {
                setLoadingShops(true);
                setShopError(null);
                try {
                    const fetchedShops = await getShops();
                    setShops(fetchedShops);
                    if (fetchedShops.length > 0) {
                        setSelectedShopId(fetchedShops[0].id);
                    }
                } catch (error) {
                    console.error("Failed to fetch shops:", error);
                    setShopError("Не вдалося завантажити список магазинів.");
                } finally {
                    setLoadingShops(false);
                }
            };
            fetchShops();
        }
    });

```

```

    }, [deliveryOption, shops.length]);

    const handleQuantityChange = (itemId, newQuantity) => {
      if (newQuantity > 0) {
        updateQuantity(itemId, newQuantity);
      }
    };

    const handlePlaceOrder = async () => {
      if (!customerPhone || !customerEmail) {
        alert('Будь ласка, заповніть контактну інформацію (телефон та email).');
        return;
      }

      if (!deliveryOption) {
        alert('Будь ласка, оберіть спосіб отримання замовлення (доставка або самовивіз).');
        return;
      }

      let orderPayload = {
        phone: customerPhone,
        email: customerEmail,
        price: total,
        is_delivery: deliveryOption === 'delivery' ? 1 : 0,
      };

      if (deliveryOption === 'delivery') {
        if (!deliveryAddress || !deliveryDate || !deliveryTime) {
          alert('Будь ласка, вкажіть адресу та бажаний час/дату доставки.');
```

```

        if (item.type === 'plant') {
            orderItem.plant_id = item.id.replace('plant_', '');
        } else if (item.type === 'bouquet') {
            orderItem.bouquet_id = item.id.replace('bouquet_', '');
        } else if (item.type === 'personal_bouquet') {
            orderItem.personal_bouquet_id =
item.id.replace('personal_bouquet_', '');
        }
        return orderItem;
    });

    try {
        if (!liqPayOrderId) {
            alert('Не вдалося отримати ідентифікатор замовлення LiqPay для
перевірки статусу. Будь ласка, спробуйте ще раз.');
```

return;

```
        }

        alert('Перевіряємо статус оплати LiqPay... Будь ласка, зачекайте до 5
хвилин.');
```

console.log(`Початок перевірки статусу оплати для LiqPay ID:
\${liqPayOrderId}`);

```

        const MAX_CHECK_TIME_MS = 5 * 60 * 1000;
        const CHECK_INTERVAL_MS = 5000;
        let paymentSuccess = false;
        const startTime = Date.now();

        while (Date.now() - startTime < MAX_CHECK_TIME_MS) {
            try {
                const statusResponse = await
paymentService.getPaymentStatus(liqPayOrderId);

                if (statusResponse.success && (statusResponse.liqpay_status ===
'success' || statusResponse.liqpay_status === 'sandbox')) {
                    paymentSuccess = true;
                    console.log('Оплата успішна:', statusResponse);
                    break;
                } else {
                    console.log(`Оплата ще не успішна. Статус:
${statusResponse.liqpay_status}. Зачекайте...`);
                }
            } catch (statusError) {
                console.warn('Помилка під час виклику getPaymentStatus:',
statusError);
            }
            await new Promise(resolve => setTimeout(resolve,
CHECK_INTERVAL_MS));
        }
    }

```

```

        if (!paymentSuccess) {
            alert('Не вдалося підтвердити оплату через LiqPay протягом 5
хвилин. Будь ласка, перевірте ваш статус оплати або зв\'яжіться зі службою
підтримки.');
```

return;

```
        }

        alert('Оплата підтверджена. Розміщуємо замовлення...');
        orderPayload.liqpay_order_id = liqPayOrderId;
        const response = await addOrder(orderPayload);

        if (response.success) {
            alert(`Замовлення успішно додано! Номер замовлення:
${response.order_id}`);
            if (clearCart) {
                clearCart();
            }
            // if (navigate) {
            //     navigate('/order-success');
            // }
        } else {
            alert(`Помилка при обробці замовлення: ${response.message ||
'Невідома помилка'}`);
        }
    } catch (error) {
        console.error('Помилка при розміщенні замовлення:', error);
        if (error.response && error.response.data &&
error.response.data.attempted_data) {
            console.error('Відправлені дані, що спричинили помилку:',
error.response.data.attempted_data);
        }
        alert('Не вдалося розмістити замовлення. Будь ласка, спробуйте ще раз
або зв\'яжіться зі службою підтримки.');
```

}

```
    };

    useEffect(() => {
        const payButton = document.getElementById('pay');
        if (payButton) {
            payButton.addEventListener('click', handlePlaceOrder);

            return () => {
                payButton.removeEventListener('click', handlePlaceOrder);
            };
        }
    }, [paymentForm, handlePlaceOrder]);

```

```

    if (cart.length === 0) {
      return (
        <div className="container py-5">
          <div className="text-center">
            <h2>Ваш кошик порожній</h2>
            <p className="text-muted">Додайте товари до кошика, щоб зробити
замовлення</p>
            <Link to="/" className="btn btn-primary mt-3">
              Повернутися до магазину
            </Link>
          </div>
        </div>
      );
    }

    return (
      <div className="container py-5">
        <h2 className="mb-4">Ваш кошик</h2>
        <div className="row">
          <div className="col-md-8">
            {cart.map((item) => (
              <div key={item.id} className="card mb-3">
                <div className="row g-0">
                  <div className="col-md-4">
                    <img
                      src={item.image}
                      className="img-fluid rounded-start"
                      alt={item.name}
                      style={{ height: '200px', objectFit:
'cover' }}
                    />
                  </div>
                  <div className="col-md-8">
                    <div className="card-body">
                      <div className="d-flex justify-content-
between align-items-start">
                        <h5 className="card-
title">{item.name}</h5>
                        <span className="badge bg-info">
                          {item.type === 'bouquet' ? 'Букет'
: 'Рослина'}
                        </span>
                      </div>
                      <p className="card-text text-
muted">{item.description}</p>
                      <div className="d-flex justify-content-
between align-items-center">
                        <div className="input-group" style={{
width: '150px' }}>
                          <button

```

```

        className="btn btn-outline-
secondary"
        onClick={() =>
handleQuantityChange(item.id, item.quantity - 1)}
        >
        -
    </button>
    <input
        type="number"
        className="form-control text-
center"
        value={item.quantity}
        onChange={(e) =>
handleQuantityChange(item.id, parseInt(e.target.value))}
        min="1"
    />
    <button
        className="btn btn-outline-
secondary"
        onClick={() =>
handleQuantityChange(item.id, item.quantity + 1)}
        >
        +
    </button>
</div>
<div className="d-flex align-items-
center">
    <h5 className="mb-0 me-
3">{item.price * item.quantity} грн</h5>
    <button
        className="btn btn-outline-
danger"
        onClick={() =>
removeFromCart(item.id)}
        >
        <i className="bi bi-trash"></i>
    </button>
</div>
</div>
</div>
</div>
</div>
    )}}
</div>
<div className="col-md-4">
    <div className="card mb-4">
        <div className="card-body">
            <h5 className="card-title">Підсумок замовлення</h5>
            <div className="d-flex justify-content-between mb-2">
                <span>Товарів:</span>

```

```

            <span>{cart.reduce((sum, item) => sum +
item.quantity, 0)} шт.</span>
        </div>
        <div className="d-flex justify-content-between mb-3">
            <span>Загальна сума:</span>
            <span className="fw-bold">{total} грн</span>
        </div>
    </div>
</div>

<div className="card mb-4">
    <div className="card-body">
        <h5 className="card-title">Контактна інформація</h5>
        <div className="mb-3">
            <label htmlFor="customerPhone" className="form-
label">Телефон:</label>

            <input
                type="tel"
                className="form-control"
                id="customerPhone"
                placeholder="+380XXXXXXXXX"
                value={customerPhone}
                onChange={(e) =>
setCustomerPhone(e.target.value)}
            />
        </div>
        <div className="mb-3">
            <label htmlFor="customerEmail" className="form-
label">Email:</label>

            <input
                type="email"
                className="form-control"
                id="customerEmail"
                placeholder="example@example.com"
                value={customerEmail}
                onChange={(e) =>
setCustomerEmail(e.target.value)}
            />
        </div>
    </div>
</div>

<div className="card mb-4">
    <div className="card-body">
        <h5 className="card-title">Оберіть спосіб отримання
замовлення</h5>

        <div className="form-check">
            <input
                className="form-check-input"
                type="radio"
                name="deliveryOption"

```

```

        id="deliveryRadio"
        value="delivery"
        checked={deliveryOption === 'delivery'}
        onChange={() => {
            setDeliveryOption('delivery');
            setSelectedShopId(null);
        }}
    />
    <label className="form-check-label"
htmlFor="deliveryRadio">
        Доставка кур'єром
    </label>
</div>
{deliveryOption === 'delivery' && (
    <div className="mt-3">
        <div className="mb-3">
            <label htmlFor="deliveryAddress"
className="form-label">
                Введіть адресу доставки:
            </label>
            <textarea
                className="form-control"
                id="deliveryAddress"
                rows="3"
                placeholder="Вулиця, будинок, квартира,
місто, поштовий індекс..."
                value={deliveryAddress}
                onChange={(e) =>
setDeliveryAddress(e.target.value)}
            ></textarea>
        </div>
        <div className="mb-3">
            <label htmlFor="deliveryDate"
className="form-label">Бажана дата доставки:</label>
            <input
                type="date"
                className="form-control"
                id="deliveryDate"
                value={deliveryDate}
                onChange={(e) =>
setDeliveryDate(e.target.value)}
            />
        </div>
        <div className="mb-3">
            <label htmlFor="deliveryTime"
className="form-label">Бажаний час доставки:</label>
            <input
                type="time"
                className="form-control"
                id="deliveryTime"
                value={deliveryTime}

```

```
onChange={(e) =>
setDeliveryTime(e.target.value)}
        />
      </div>
    </div>
  )}
<div className="form-check mt-2">
  <input
    className="form-check-input"
    type="radio"
    name="deliveryOption"
    id="pickupRadio"
    value="pickup"
    checked={deliveryOption === 'pickup'}
    onChange={() => {
      setDeliveryOption('pickup');
      setDeliveryAddress('');
      setDeliveryDate('');
      setDeliveryTime('');
    }}
  />
  <label className="form-check-label"
htmlFor="pickupRadio">
    Самовивіз з магазину
  </label>
</div>
{deliveryOption === 'pickup' && (
  <div className="mt-3">
    {loadingShops ? (
      <div className="text-center">
        <div className="spinner-border text-
primary" role="status">
          <span className="visually-
hidden">Завантаження...</span>
        </div>
        <p className="mt-2">Завантаження
магазинів...</p>
      </div>
    ) : shopError ? (
      <div className="alert alert-danger"
role="alert">
        {shopError}
      </div>
    ) : (
      <>
        {shops.length > 0 ? (
          <div className="list-group">
            <p className="text-muted mb-
2">Оберіть магазин:</p>
            {shops.map((shop) => (
```

```

        className="list-group-item">

        check-input me-1"

        checked={selectedShopId === shop.id}

        setSelectedShopId(shop.id)}

        <label key={shop.id}

        <input

        className="form-

        type="radio"

        name="selectedShop"

        value={shop.id}

        onChange={() =>

        />

        {shop.address}

        </label>

        )}}

        </div>

        ) : (

        <p className="text-muted">Наразі

        немає доступних магазинів для самовивозу.</p>

        )}

        </>

        )}

        </div>

        )}

        </div>

        </div>

        {paymentForm && (

        <div className="card">

        <div className="card-body">

        <h5 className="card-title mb-3">Оплата

        замовлення</h5>

        <div dangerouslySetInnerHTML={{ __html: paymentForm

        }} />

        </div>

        </div>

        )}

        </div>

        </div>

        </div>

        );

    };

    export default CartPage;

<?php

namespace core;

use controllers\MainController;

```

Додаток Е

```

use Exception;
use core\DBContext;

include('config/database.php');

class Core
{
    private static $instance = null;

    public $context;
    public $app;

    private function __construct()
    {
    }

    public static function getInstance()
    {
        if (empty(self::$instance)) {
            self::$instance = new self();
        }
        return self::$instance;
    }

    public function Redirect($where)
    {
        header("Location: $where");
    }

    public function Initialize()
    {
        session_start();
        $this->context = new DBContext(DATABASE_HOST, DATABASE_BASENAME,
DATABASE_LOGIN, DATABASE_PASSWORD);
    }

    public function Run()
    {
        $route = $_GET["route"];
        $routeParts = explode('/', $route);

        $moduleName = array_shift($routeParts);
        $actionName = array_shift($routeParts);
        if (empty($moduleName)) {
            $moduleName = "main";
        }
        if (empty($actionName)) {
            $actionName = "index";
        }

        $controllerName = "\\controllers\\" . ucfirst($moduleName) . "Controller";
        $controllerActionName = $actionName . "Action";
    }
}

```

```

try {
    if (!class_exists($controllerName)) {
        throw new Exception("Not found", 404);
    }
    $controller = new $controllerName();
    if (!method_exists($controller, $controllerActionName)) {
        throw new Exception("Not found", 404);
    }

    $this->app["moduleName"] = $moduleName;
    $this->app["actionName"] = $actionName;

    $this->app["content"] = $controller->$controllerActionName($routeParts);
} catch (Exception $ex) {
    $mainController = new MainController();
    $this->app["content"] = $mainController->errorAction($ex->getMessage(),
$ex->getCode());
}
}
public function Done()
{
    if($_SESSION['theme'] == 'black'){
        $layoutPath = "themes\dark\layout.php";
    }else{
        $layoutPath = "themes\light\layout.php";
    }
    $tpl = new Template($layoutPath);
    $tpl->addParam("content", $this->app["content"]);
    echo $tpl->getHTML();
}
}

```

Додаток F

```

<?php

namespace core;

class Controller {

    public function render($params = [], $viewPath = null) {
        if ($viewPath == null) {
            $moduleName = Core::getInstance()->app["moduleName"];
            $actionName = Core::getInstance()->app["actionName"];

            $viewPath = "views/$moduleName/$actionName.php";
        }
        $tpl = new Template($viewPath);
        $tpl->addParams($params);
        return $tpl->getHTML();
    }
}

```

```

}

<?php

namespace models;

use core\Core;

class Bouquet
{
    public static $tableName = "bouquet";

    public static function getBouquet()
    {
        return Core::getInstance()->context->table(self::$tableName)->select()->execute();
    }

    public static function getBouquetById($id)
    {
        $result = Core::getInstance()->context->table(self::$tableName)->select()->where([
            "id" => $id
        ])->execute();
        return $result[0] ?? null;
    }

    public static function addBouquet($name, $image, $price)
    {
        $context = Core::getInstance()->context;
        $result = $context->table(self::$tableName)
            ->insert([
                'name' => $name,
                'image' => $image,
                'price' => $price
            ])
            ->execute();

        if ($result) {
            return $context->lastInsertId();
        }
        return false;
    }

    public static function updateBouquet($id, $name, $image, $price)
    {
        return Core::getInstance()->context->table(self::$tableName)
            ->update([
                'name' => $name,
                'image' => $image,
            ])
            ->execute();
    }
}

```

```

        'price' => $price
    ])
    ->where(['id' => $id])
    ->execute();
}

public static function deleteBouquet($id)
{
    return Core::getInstance()->context->table(self::$tableName)
        ->delete()
        ->where(['id' => $id])
        ->execute();
}

public static function getFilteredBouquets(
    $searchTerm = null,
    $minPrice = null,
    $maxPrice = null,
    $requiredPlantIds = []
){
    $query = Core::getInstance()->context->table(self::$tableName)->select();
    $conditions = [];

    if ($searchTerm) {
        $query->where("name LIKE ?", ['%' . $searchTerm . '%']);
    }

    if ($minPrice !== null && $minPrice !== '') {
        $query->andWhere("price >= ?", [$minPrice]);
    }
    if ($maxPrice !== null && $maxPrice !== '') {
        $query->andWhere("price <= ?", [$maxPrice]);
    }

    if (!empty($requiredPlantIds)) {
        $subQueryConditions = [];
        foreach ($requiredPlantIds as $plantId) {
            $subQueryConditions[] = "EXISTS (SELECT 1 FROM bouquet_flower bf
WHERE bf.bouquet_id = " . self::$tableName . ".id AND bf.plant_id = $plantId)";
        }
        if (!empty($subQueryConditions)) {
            $query->andWhere(implode(' AND ', $subQueryConditions));
        }
    }

    return $query->execute();
}
}

```

```

<?php

namespace core;

use PDO;

class DBContext {
    protected $connection;

    public function __construct($host, $dbname, $login, $password) {
        $this->connection = new PDO("mysql: host=$host;dbname=$dbname", $login,
$password);
    }

    public function lastInsertId($name = null){
        return $this->connection->lastInsertId($name);
    }

    public function table($tableName){
        $query = new QueryBuilder($this->connection, $tableName);
        return $query;
    }
}

```

Додаток I

```

<?php

namespace core;

use PDO;

class QueryBuilder {
    protected $connection;

    protected $tableName;
    protected $statementParams;

    protected $primaryPart;
    protected $wherePart;
    protected $orderByPart;

    public function __construct(PDO $connection, $tableName) {
        $this->connection = $connection;
        $this->tableName = $tableName;
        $this->statementParams = [];
        $this->primaryPart = '';
        $this->wherePart = '';
        $this->orderByPart = '';
    }

    public function select($fields = "*") {

```

```

        if (is_array($fields)) {
            $fields = implode(', ', $fields);
        }
        $this->primaryPart = "SELECT $fields FROM $this->tableName ";

        return $this;
    }

    public function insert($params) {
        $fields = implode(", ", array_keys($params));
        $valuesParts = [];
        foreach ($params as $key => $value) {
            $paramKey = ":insert$key";
            $this->statementParams[$paramKey] = $value;
            $valuesParts[] = $paramKey;
        }
        $values = implode(", ", $valuesParts);
        $this->primaryPart = "INSERT INTO $this->tableName ($fields) VALUES
($values) ";
        return $this;
    }

    public function update($params) {
        $valuesParts = [];
        foreach ($params as $key => $value) {
            $param = ":update$key";
            $this->statementParams[$param] = $value;
            $valuesParts[] = "$key = $param";
        }
        $values = implode(", ", $valuesParts);
        $this->primaryPart = "UPDATE $this->tableName SET $values ";
        return $this;
    }

    public function delete() {
        $this->primaryPart = "DELETE FROM $this->tableName ";
        return $this;
    }

    public function where($params, $conditionValues = []) {
        $conditionParts = [];
        if (is_array($params)) {
            foreach ($params as $key => $value) {
                $param = ":where_and_" . str_replace('.', '_', $key);
                $this->statementParams[$param] = $value;
                $conditionParts[] = "$key = $param";
            }
            $condition = implode(" AND ", $conditionParts);
        } else {
            $condition = $params;
        }
    }

```

```

        $this->statementParams = array_merge($this->statementParams,
$conditionValues);
    }

    $this->wherePart = "WHERE $condition ";
    return $this;
}

public function andWhere($params, $conditionValues = []) {
    if (empty($this->wherePart)) {
        return $this->where($params, $conditionValues);
    }

    $conditionParts = [];
    if (is_array($params)) {
        foreach ($params as $key => $value) {
            $param = ":and_where_" . str_replace('.', '_', $key) . uniqid();
            $this->statementParams[$param] = $value;
            $conditionParts[] = "$key = $param";
        }
        $condition = implode(" AND ", $conditionParts);
    } else {
        $condition = $params;
        $this->statementParams = array_merge($this->statementParams,
$conditionValues);
    }

    $this->wherePart .= "AND $condition ";
    return $this;
}

public function orderBy($fields) {
    if (is_array($fields)) {
        $fields = implode(', ', $fields);
    }
    $this->orderByPart = "ORDER BY $fields ";
    return $this;
}

public function execute() {
    $sql = $this->primaryPart . $this->wherePart . $this->orderByPart;
    $result = $this->connection->prepare($sql);

    $execResult = $result->execute($this->statementParams);
    $fetchResult = $result->fetchAll(PDO::FETCH_ASSOC);

    $this->primaryPart = '';
    $this->wherePart = '';
    $this->orderByPart = '';
    $this->statementParams = [];
}

```

```

        if (strpos($this->primaryPart, "SELECT") !== false || strpos($sql,
"SELECT") !== false) { // Check both for robustness
            return $fetchResult;
        }
        return $execResult;
    }
}

```

Додаток J

```

<?php
define('DATABASE_HOST', 'localhost');
define('DATABASE_LOGIN', 'root');
define('DATABASE_PASSWORD', '');
define('DATABASE_BASENAME', 'flower_shop');
?>
<?php

define('LIQPAY_PUBLIC_KEY', 'sandbox_i73315462191');
define('LIQPAY_PRIVATE_KEY', 'sandbox_EGbgVDxjGWNT1woxNPwBotApChT3Uh2Cdq2nYknb');
<?php
define('MAILER_HOST', 'smtp.gmail.com');
define('MAILER_SMTPAuth', true);
define('MAILER_EMAIL', 'animeshnich007@gmail.com');
define('MAILER_KEY', 'inzqlipsggptwxeq');
define('MAILER_SMTPSecure', 'ssl');
define('MAILER_PORT', 465);

```