

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:

«Онлайн-сервіс замовлення та доставки квітів»

Виконала студентка групи ПЗк-21-1
КОЛОСОВИЧ Олександра Олександрівна

Керівник роботи:
КОНЦЕДАЙЛО Валерій Валерійович

Консультант:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2024

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

«ЗАТВЕРДЖУЮ»

Завідувач кафедри інженерії
програмного забезпечення

Тетяна ВАКАЛЮК

«23» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **КОЛОСОВИЧ Олександра Олександрівна**

Керівник роботи: **КОНЦЕДАЙЛО Валерій Валерійович**

Тема роботи: **«Онлайн-сервіс замовлення та доставки квітів»**,

затверджена Наказом закладу вищої освіти від **«23» лютого 2024 р., №74/с**

Вихідні дані для роботи: розробка онлайн сервісу замовлення та доставки квітів

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Концедайло В.В. Чижмотря О.В.	25.02.2024р.	25.02.2024р.
2	Концедайло В.В. Чижмотря О.В.	25.02.2024р.	25.02.2024р.
3	Концедайло В.В. Чижмотря О.В.	25.02.2024р.	25.02.2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	25 лютого 2024 – 29 лютого 2024	Виконано
2	Аналіз моделей аналогів	1 березня 2024 – 9 березня 2024	Виконано
3	Постановка технічного завдання	10 березня 2024 – 14 березня 2024	Виконано
4	Вивчення літературних джерел	15 березня 2024 – 20 березня 2024	Виконано
5	Проектування структури	21 березня 2024 – 25 березня 2024	Виконано
6	Написання програмного коду	26 березня 2024 – 18 квітня 2024	Виконано
7	Тестування роботи програми	19 квітня 2024 – 23 квітня 2024	Виконано
8	Аналіз одержаних результатів	24 квітня 2024 – 26 квітня 2024	Виконано
9	Оформлення пояснювальної записки	27 квітня 2024 – 25 травня 2024	Виконано
10	Попередній захист	1 травня 2024	Виконано

РЕФЕРАТ

Випускна робота бакалавра складається з комплексної програмної реалізації онлайн сервісу замовлення та доставки квітів, а також пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить 87 сторінок, 43 ілюстрації, 11 таблиць, 9 додатків та 30 літературних джерел.

Основною метою даної роботи є розробка та впровадження програмного продукту, який дозволяє користувачам зручно замовляти та отримувати квіти через веб-сайт. Програмний продукт включає різні функціональні можливості для адміністраторів та продавців-флористів, зокрема управління замовленнями, додавання нових букетів та категорій, а також забезпечення безпеки та продуктивності системи.

В роботі визначено ключові завдання, що пов'язані з розробкою сервісу, проведено аналіз існуючих рішень у сфері онлайн-замовлення квітів, а також наведено сценарії використання програмного продукту. Представлено огляд загальної структури додатку, описано алгоритми взаємодії між окремими компонентами, а також наведено опис об'єктної структури (діаграму класів) додатку.

В рамках випускної роботи проводились експерименти по оптимізації роботи з базою даних та інтерактивними елементами інтерфейсу, в результаті яких були визначені оптимальні налаштування для швидкої та надійної роботи сервісу.

КЛЮЧОВІ СЛОВА: ОНЛАЙН-СЕРВІС, ЗАМОВЛЕННЯ КВІТІВ, PHP, MySQL, HTML, CSS, JAVASCRIPT, ВЕБ-РОЗРОБКА.

					ІПЗ.КР.Б – 121 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Онлайн-сервіс замовлення та доставки квітів Пояснювальна записка	Літ.	Арк.	Аркушів
Розроб.		О.О. Колосович						
Керівник		В.В. Концедайло					4	87
Реценз.		Ю.М. Єфремов				Житомирська політехніка, група ІПЗк-21-1		
Затверд.		Т.А. Вакалюк						

ABSTRACT

The bachelor's thesis consists of a complex software implementation of an online flower ordering and delivery service, as well as an explanatory note. The explanatory note to the thesis contains 87 pages, 43 illustrations, 11 tables, 9 appendices and 30 literary sources.

The main goal of this work is the development and implementation of a software product that allows users to conveniently order and receive flowers through the website. The software product includes various functionalities for administrators and florists, including managing orders, adding new bouquets and categories, and ensuring system security and performance.

The paper defines the key tasks related to the development of the service, analyzes existing solutions in the field of online flower ordering, and also provides scenarios for the use of the software product. An overview of the general structure of the application is presented, the algorithms of interaction between individual components are described, and a description of the object structure (class diagram) of the application is also given.

As part of the final work, experiments were conducted to optimize work with the database and interactive interface elements, as a result of which the optimal settings for fast and reliable service operation were determined.

KEYWORDS: ONLINE SERVICE, ORDER FLOWERS, PHP, MySQL, HTML, CSS, JAVASCRIPT, WEB DEVELOPMENT.

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		5

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ.....	10
1.1. Постановка задачі	10
1.2. Аналіз існуючого програмного забезпечення	12
1.3. Обґрунтування вибору інструментальних засобів.....	16
Висновки до першого розділу.....	19
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	21
2.1. Технічне завдання на розробку системи.....	21
2.2. Аналіз вимог до програмного продукту	22
2.3. Алгоритм роботи та стани програмної системи.....	28
2.4. Об'єктно-орієнтована модель системи	30
2.5. Взаємодія об'єктів системи	35
2.6. Розробка бази даних системи	37
Висновки до другого розділу	42
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ	44
3.1. Порядок встановлення та налаштування системи	44
3.2. Розробка основних функцій магазину	46
3.3. Опис інструкції користувача	52
3.4. Тестування роботи програмного продукту	63
Висновки до третього розділу.....	66
ВИСНОВКИ.....	67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	68
ДОДАТКИ	71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AJAX – Asynchronous Javascript and XML;

CSS – Cascading Style Sheets;

HTML – HyperText Markup Language;

MVC – model-view-controller;

PHP – hypertext preprocessor;

SQL – structured query language;

UML – Unified Modeling Language;

XML – Extensible Markup Language;

БД – база даних;

ІМ – інтернет-магазин;

ПЗ – програмне забезпечення;

СУБД – система управління базами-даних.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

ВСТУП

Актуальність теми. У сучасному світі інформаційних технологій та електронної комерції, сервіси онлайн доставки стали невід'ємною складовою життя суспільства. Їх важливість постійно зростає, забезпечуючи зручність та ефективність у різних галузях. Однією з активно розвиваючихся сфер в цьому контексті є сервіси доставки квітів, які відкривають нові можливості для споживачів та флористичних підприємств [6].

Система автоматизованого підбору квітів є програмним інструментом, який сприяє знаходженню ідеального букету квітів відповідно до вимог та переваг користувачів. Вона включає в себе методи та алгоритми для класифікації параметрів клієнта з метою отримання оптимального результату. Загалом, автоматизована система підбору букетів квітів спрощує процес вибору та замовлення квітів, забезпечуючи зручність, швидкість і точність.

Основною метою цієї системи є оптимізація процесу обробки параметрів, який включає такі етапи:

- Отримання параметрів: Вони передаються у програму через веб-запит користувача, важливо забезпечити коректне та швидке отримання цих параметрів.
- Перевірка та валідація: Параметри перевіряються на правильність та валідність, включаючи перевірку діапазону значень та наявність обов'язкових параметрів.
- Обробка: Параметри обробляються відповідно до їх значень та особливостей, включаючи встановлення налаштувань, виклик певних функцій, виконання дій та встановлення змінних.
- Використання оброблених параметрів: Оброблені параметри використовуються для подальшої роботи системи, зокрема, для сортування товарів та виведення підходящих букетів клієнту.
- Реалізація цих етапів сприяє зменшенню часу, необхідного для вибору товару, оскільки клієнт вже має зменшену вибірку. Автоматизація процесу

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		8

підбору також покращує обслуговування клієнтів, забезпечуючи швидку та точну обробку замовлень і надаючи персоналізовані рекомендації, що допомагає уникнути людських помилок.

Більшість існуючих систем квітів є обмеженими або не мають параметрів підбору, що призводить до додаткових зусиль та часу витрачених на пошук. Автоматизована система також допомагає іншим працівникам зосередитися на інших справах, підвищуючи загальну продуктивність.

Мета даної кваліфікаційної роботи полягає в розробці алгоритму для автоматизованої системи підбору букетів, а для досягнення цієї мети були сформульовані наступні завдання:

- а) Аналіз існуючих рішень та програмних застосунків, конкурентів та успішних прикладів з інших галузей.
- б) Формулювання вимог до функціоналу майбутньої системи.
- в) Розробка технічного завдання, алгоритму та схем відповідно до сформульованого опису функціоналу.
- г) Реалізація функціоналу автоматизованої системи.

Об'єктом дослідження є автоматизовані системи підбору та обробки товарів, а предметом - сучасні технології автоматизації підбору товарів.

Предметом дослідження є онлайн-сервіс замовлення та доставки квітів.

Методи дослідження включають порівняльний аналіз інших магазинів, аналіз літературних джерел, вивчення структури та дизайну сайтів, моделювання поведінки користувачів, використання мов програмування з орієнтацією на об'єкти, таких як JavaScript для клієнтської частини та PHP для серверної частини, а також тестування різних варіантів взаємодії з користувачами.

Практична значимість кваліфікаційної роботи полягає в тому, щоб забезпечити зручність для користувачів, автоматизацію процесів, розширення ринкового охоплення, покращення обслуговування клієнтів, аналіз даних для маркетингу та гнучкість системи.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1. Постановка задачі

Мета розроблюваного проекту полягає у створенні функціоналу для запуску та постійної підтримки інтернет-магазину з доставкою квітів, а також забезпечення користувачів інструментами для вибору, замовлення та обробки букетів квітів. Реалізація магазину дозволить клієнтам замовляти квіти за допомогою онлайн-платформи, вибираючи букети за розміром, видом квітів, оформленням та іншими параметрами; редагувати замовлення, змінюючи склад, розмір або додаванням додаткових послуг.

Основні вимоги до веб-сайту визначаються наступним чином:

1. Дизайн сайту має відповідати світлій кольоровій палітрі, з дотриманням фірмового стилю та естетики.
2. Забезпечення максимальної швидкості доступу до інформації, забезпечуючи мінімальний час завантаження сторінок.
3. Організація навігації по сайту максимально зручною та зрозумілою для різних категорій відвідувачів, як для досвідчених, так і для новачків в Інтернет-технологіях.
4. Забезпечення зручного та інтуїтивно зрозумілого інтерфейсу користувача.

Адміністративна панель, доступ до якої здійснюється через авторизацію, повинна відповідати наступним вимогам:

1. Можливість додавання та редагування сезонних пропозицій.
2. Можливість додавання та редагування каталогу букетів.
3. Можливість редагування найкращих пропозицій. Можливість видалення товарів.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		10

Для досягнення цієї мети проект передбачає такі етапи:

1. Аналіз та визначення вимог:

- 1.1. дослідження практики використання інтернет-магазинів доставки квітів.
- 1.2. визначення основних характеристик та функціональних можливостей магазину.
- 1.3. аналіз конкурентів та аналогів програмного продукту.

2. Вибір технологій та інтеграція:

- 2.1. вибір платформи розробки.
- 2.2. інтеграція стандартних інструментів та фреймворків для роботи магазину.
- 2.3. розгортання програмного продукту для роботи на онлайн-платформі.

3. Розробка та впровадження:

- 3.1. розробка інтуїтивно зрозумілого та зручного інтерфейсу магазину.
- 3.2. реалізація основних функцій, таких як вибір та замовлення квітів, редагування замовлення, оплата та доставка.
- 3.3. реалізація можливостей редагування замовлень, таких як відміна та зміна деталей замовлення.

4. Тестування та забезпечення якості:

- 4.1. проведення ретельного тестування для перевірки роботи програмного забезпечення та відповідності вимогам.
- 4.2. забезпечення якості розробки та тестування.

5. Розгортання та підтримка:

- 5.1. розгортання програмного рішення для користувачів.
- 5.2. забезпечення підтримки та обслуговування після впровадження.

Всі дані, введені через адміністративну панель, повинні автоматично заноситися в базу даних інтернет-магазину та відображатися на клієнтській частині сайту за допомогою створених шаблонів даних. Успішна розробка та

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		11

впровадження інтернет-магазину доставки квітів надасть клієнтам можливість швидко та зручно замовляти квіти через Інтернет, що значно спростить процес вибору та зробить його більш доступним для широкого кола користувачів.

1.2. Аналіз існуючого програмного забезпечення

На сучасному ринку існує ряд програмних рішень, спрямованих на автоматизацію та оптимізацію процесів у сфері онлайн доставки квітів. Аналіз існуючого програмного забезпечення дозволяє виділити певні тенденції та особливості цього сегменту. Опишемо детально знайдені реалізовані рішення.

Flowers De Lux – <https://flowersdelux.com.ua>

Сайт "Flowers De Lux" (рис. 1.1) є віртуальною студією флористики, яка спеціалізується на продажу не лише квітів, але й різноманітних доповнень до них, таких як гелієві кульки, піньяти, фігурки з квітів, фруктові та шоколадні корзини, а також подарунки [13].

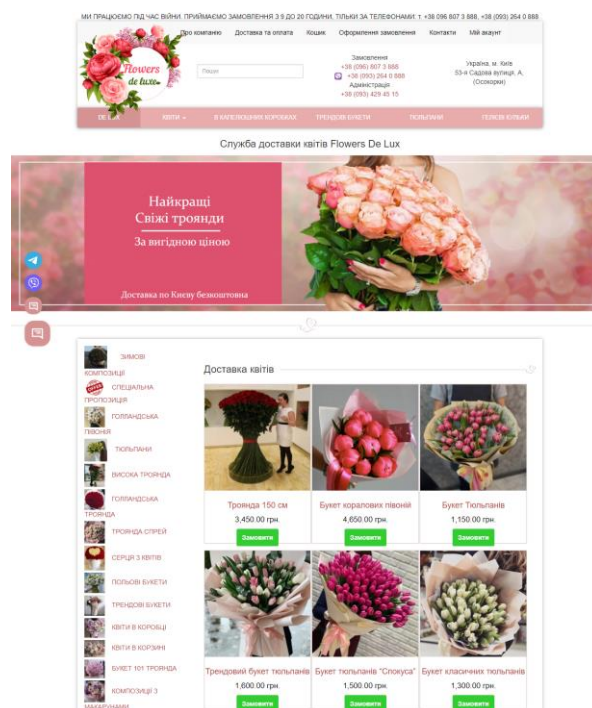


Рисунок 1.1 – Головна сторінка сайту Flowers De Lux

На цьому сайті доступні лише готові букети з зазначеною кількістю квітів, і немає можливості створювати свій букет. Вартість букетів зазначена напряму

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

на сторінці товарів за мінімальну кількість квітів, також ціни можна дізнатися, перейшовши на сторінку конкретного товару та обравши бажану кількість квітів.

Сайт дозволяє застосовувати фільтри до обраної категорії за такими критеріями, як популярність, рейтинг, час внесення та ціна.

Інформація про доставку та оплату ретельно представлена на відповідній сторінці, а контактна інформація легко доступна як на головній сторінці, так і на сторінці "Контакти". Сторінка "Про компанію" містить лише текстову інформацію і не має фотогалереї магазину (рис. 1.2).

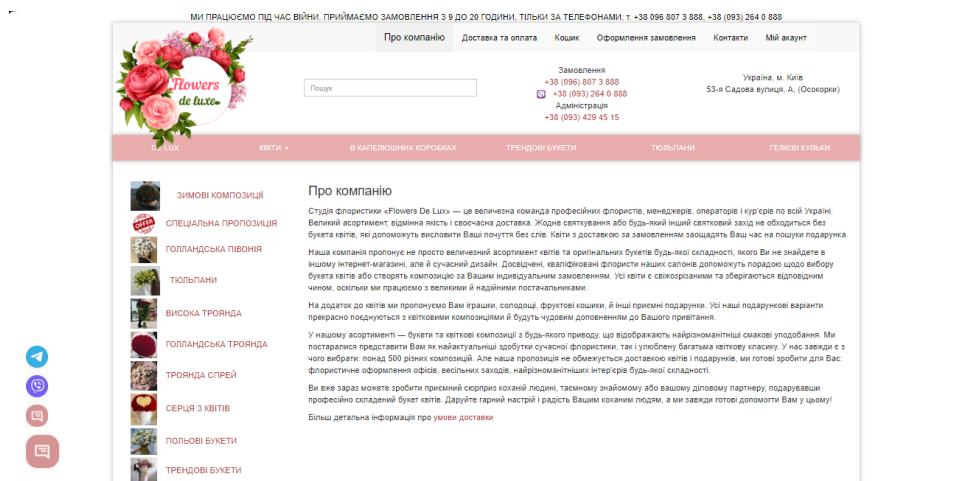


Рисунок 1.2 – Сторінка «Про компанію» сайту Flowers De Lux

Є можливість входу в особистий кабінет користувача, але відсутні сторінки або кнопки реєстрації для нових користувачів, що може ускладнити процес створення акаунта.

Bloom – <https://bloom.ua>

Сайт "Bloom" представляє квіткову майстерню, яка запропонувала широкий вибір продукції, включаючи різноманітні букети, окремі квіти, квіткові композиції, рослини в горщиках та різноманітні подарунки, такі як листівки, повітряні кульки, солодощі, м'які іграшки, подарункові сертифікати, кераміку та вази [22].

Дизайн сайту вражає своєю естетичністю і професіоналізмом, він має сучасний вигляд та зручний інтерфейс. Меню та фільтри дозволяють легко сортувати товари за ціною, наявністю, категорією, підкатегорією, типами квітів

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

і розміром букетів. Кожен товар має якісні зображення, ціну, артикул та кнопку "В кошик". Однак картка товару може перекривати зображення, що може впливати на загальний вигляд.

Інформація про доставку та оплату надана на окремій сторінці, де клієнти можуть обрати дату та час доставки, та обчислити вартість доставки, скориставшись формою на сторінці "Оплата і Доставка" (рис. 1.3).

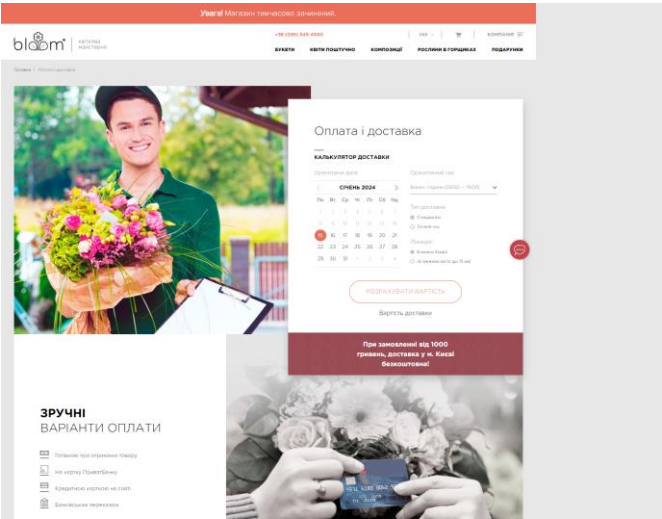


Рисунок 1.3 – Сторінка «Оплата і доставка» сайту «Bloom»

На сайті відсутня інформація про сам магазин, за винятком графіка роботи, контактних даних, мапи місцезнаходження та електронної адреси. Бракує фотогалереї магазину та сторінки відгуків, які можуть сприяти покращенню довіри користувачів.

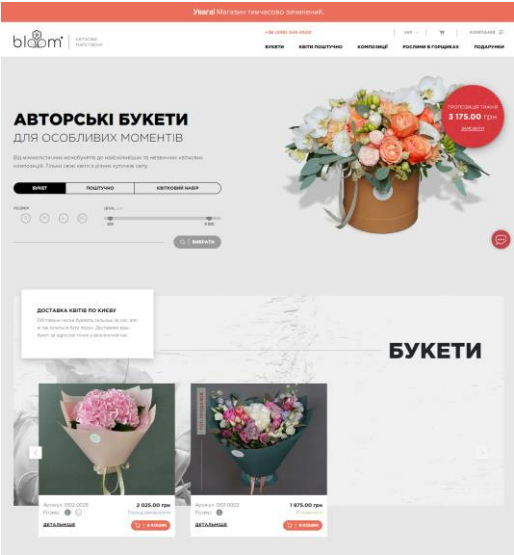


Рисунок 1.4 – Головна сторінка сайту «Bloom»

Flora24 – flora24.com.ua

Сайт "Flora24.com.ua" (рис. 1.5) вражає своєю непримхливістю та відсутністю сучасного дизайну, не витягаючи на себе увагу [5].

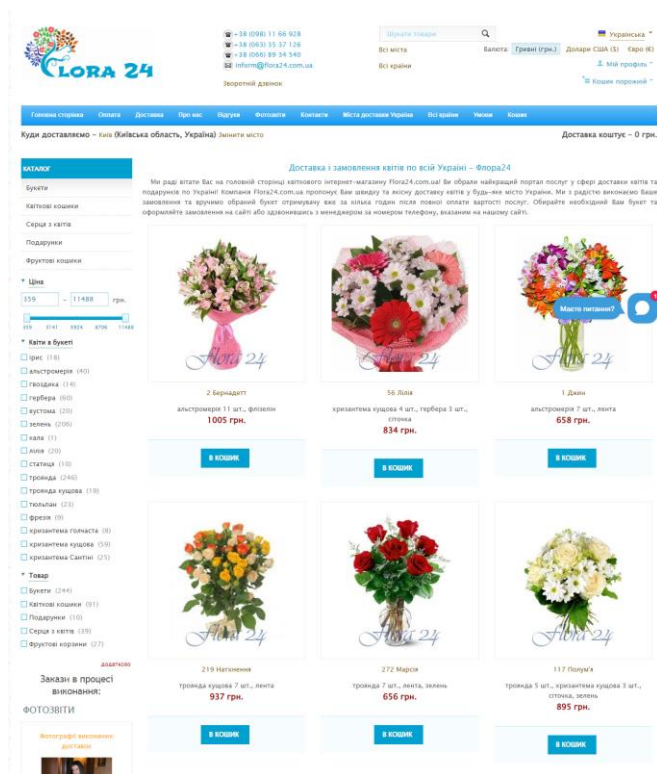


Рисунок 1.5 – Головна сторінка сайту «Flora24

Магазин пропонує різноманітні товари, такі як букети, квіткові кошики, серця з квітів, подарунки (м'які іграшки та солодоші) та фруктові кошики. Інтерфейс сайту підтримує дві мови: українську та російську, і їх відображення коректне.

Зображення товарів може бути удосконалене, хоча картка товару містить важливу інформацію та переваги магазину. Проте обмеженість у виборі квітів та їх кількості в букеті може відлякати покупців.

Інформаційні сторінки "Оплата" і "Доставка" корисні, але вартість доставки не вказана. Інформації про сам магазин обмежено, але вона добре сприймається, створюючи враження надійності та впевненості.

Сайт має сторінку відгуків з численними позитивними відгуками та фотозвітами виконаних замовлень, що може бути привабливим для нових клієнтів. Наявні контактні дані і зручний кабінет користувача з реєстрацією,

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		15

входом та можливістю перегляду історії замовлень роблять сайт функціональним і зручним для клієнтів.

Таблиця 1.1

Порівняння існуючих аналогів

Критерії	Flowers De Lux	Bloom	Flora24	Floral Paradise
Занесення товарів у список обраного	+	+	+	+
Порівняння товарів	—	+	-	+
Перевантаженість інтерфейсу	+	—	+	—
Перевантаженість інтерфейсу	+	—	+	—

1.3. Обґрунтування вибору інструментальних засобів

Обґрунтування вибору інструментальних засобів є важливим етапом у розробці будь-якого програмного продукту чи веб-сайту.

Для створення клієнтської частини Інформаційної системи використовувалися такі технології розробки та мови програмування front-end:

- HTML;
- CSS;
- JavaScript;
- JQuery;
- Ajax.

Цей набір технологій є відмінною класичною вибіркою для розробки веб-сайтів, оскільки він продемонстрував свою надійність. Основні переваги цього набору полягають у ефективній та стабільній роботі системи та високій

швидкодії операцій як для користувача-клієнта, так і для користувача-адміністратора.

HTML є мовою гіпертекстової розмітки сторінок і використовується для визначення того, як браузер повинен відобразити завантажений сайт. Принцип роботи розмітки полягає в тому, що спочатку вводиться адреса сайту, браузер робить запит за цією адресою і отримує файл у форматі .html. Код з отриманого файлу перетворюється у візуальні об'єкти, дотримуючись правил перетворення, які можуть відрізнятися в залежності від системи, версії та типу браузера [11].

CSS є формальною мовою для опису оформлення зовнішнього вигляду сторінки, створеної за технологіями мови розмітки (HTML, XML, XHTML). Відмінною рисою CSS є можливість вибіркового застосування стилів до окремих елементів у документах типу HTML, забезпечуючи гнучкість, простий код та можливість використовувати готові бібліотеки з кодом [15].

JavaScript - це популярна мова програмування, яка дозволяє створювати складні рішення на веб-сторінках, покращуючи інтерактивність та швидкість обробки дій користувачів. З особливим попитом JavaScript користується з виникненням технології AJAX, яка дозволяє асинхронно оновлювати веб-сторінку без перезавантаження всієї сторінки [14].

jQuery - це бібліотека, яка надає високий рівень абстракції для вирішення типових задач веб-розробки. Використовуючи jQuery, можна легко застосовувати розширені можливості в будь-яких ситуаціях, що значно полегшує розробку [3].

AJAX використовується для асинхронного доступу до веб-серверів з веб-сторінки, дозволяючи оновлювати лише певні частини сторінки без перезавантаження її повністю [3].

Серверна частина Інформаційної системи використовує наступні технології:

- PHP (Hypertext Preprocessor);
- SQL (Structured Query Language).

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

PHP (розшифровується як Hypertext Preprocessor) представляє собою одну з мов сценаріїв, яка використовується наряду з вищезначеною мовою JavaScript. Основна відмінність між цими двома мовами полягає в тому, що PHP призначений для використання на стороні сервера, тоді як JavaScript може використовуватися як для фронтенд-розробки, так і на стороні бекенд. Таким чином, PHP в змозі обробляти різноманітні завдання на стороні сервера, такі як збір даних з форм, зміна баз даних та управління файлами на сервері [10], [27].

SQL представляє собою інструмент, створений для організації, управління, обробки та вибірки інформації, що зберігається в базі даних. Незважаючи на те, що вибірка даних є однією з основних функцій SQL, на сьогодні ця мова застосовується для реалізації всіх можливостей, які може надати система управління базами даних (СУБД). Це включає в себе можливість визначення користувачем організації та структури даних, їх вилучення та обробку, а також внесення змін до БД, забезпечення захисту від несанкціонованого доступу та забезпечення відмовостійкості за рахунок транзакцій та інших засобів.

Отже, SQL виконує роль потужної мови для управління СУБД та взаємодії з нею, надаючи декларативні інструкції, спрямовані на вирішення завдань управління базами даних.

Для обробки і зберігання інформації зазвичай використовують MySQL. MySQL – це система управління реляційними базами даних. У реляційній базі даних дані зберігаються в окремих таблицях, завдяки чому досягається виграш у швидкості й гнучкості. Таблиці зв'язуються між собою за допомогою відносин, завдяки чому забезпечується можливість поєднувати при виконанні запиту дані з декількох таблиць. SQL як частину системи MySQL можна охарактеризувати як мову структурованих запитів, що використовується для доступу до баз даних [9].

Висновки до першого розділу

У першому розділі мого дослідження був проведений аналіз предмета вивчення. Під час цього аналізу були виявлені труднощі, які виникають у наявних реалізаціях, а також були визначені переваги і недоліки цих реалізацій.

На основі проведеного аналізу були сформульовані конкретні вимоги до автоматизованої системи вибору букетів квітів і визначені завдання, які потрібно вирішити під час подальшого розроблення. Аналіз об'єкта дослідження допоміг з'ясувати, які саме проблеми потрібно вирішити за допомогою автоматизованої системи вибору букетів квітів.

Було розглянуто концепції представницьких веб-сайтів та інтернет-магазинів, вивчені їхні особливості та компоненти. Проведено аналіз трьох веб-сайтів, представницьких магазинів квітів: flowersdelux.com.ua, bloom.ua та flora24.com.ua.

Під час дослідження цих сайтів було виявлено наступні недоліки:

1. На деяких сайтах спостерігаються елементи, розміщені неправильно, що призводить до їхнього перекриття та взаємодії з іншими об'єктами. Це створює враження не професійності та заважає користувачам зручно переглядати товар та його характеристики.
2. Більшість розглянутих магазинів не надають можливість придбання окремих квітів у роздріб, відсутні параметри та опис для кожного виду квітки. Це значущий недолік, оскільки відсутність інформації про товар може впливати на рішення покупця, особливо при онлайн-покупках, де немає можливості фізично оглянути товар.
3. Розглянуті сайти мають обмежену кількість інформації щодо доставки, відомостей про магазин, його переваг та фотогалерей. Ця інформація впливає на перше враження від магазину, довіру до нього та упевненість у якості наданих продуктів.

4. У двох магазинах зазначеної вибірки є розділ "Мій акаунт", але на одному з сайтів виявлено, що форма для реєстрації в цей акаунт відсутня. Це викликає сумніви у функціональності цього розділу та можливості користування ним.

Основною перевагою запропонованого проекту буде не лише можливість представлення магазину, його просування та ознайомлення з асортиментом, але й можливість детального перегляду товарів, їх параметрів, відгуків та замовлення не лише готових букетів, а й персоналізованих композицій для рідних та близьких.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		20

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Технічне завдання на розробку системи

1) Назва системи: Квітковий магазин «Floral Paradise».

Повне найменування системи: Онлайн сервіс замовлень та доставки квітів.

Коротке найменування системи: Програма, Програмне забезпечення (ПЗ), Програмний комплекс, Система.

2) Призначення системи:

Програмний комплекс має на меті надати зручний та ефективний інструмент для користувачів, щоб вони могли легко та швидко замовляти та отримувати квіти за допомогою Інтернету.

3) Вимоги до системи:

Програмний продукт повинен мати наступні функціональні характеристики:

- Доступ до системи за допомогою звичних для сучасної людини браузерів;
- Модуль для створення та управління акціями, знижками та промокодами.
- Зручний інструмент для додавання, редагування та видалення товарів з каталогу.
- Можливість одночасної роботи з системою як мінімум 50 користувачів;
- Наявність двох ролей користувачів (адміністратор та зареєстрований користувач).

4) Вимоги до інтерфейсу:

Взаємодія користувачів із програмним комплексом повинна відбуватися за допомогою веб-інтерфейсів, а точніше веб-браузерів. Інтерфейс програмного комплексу повинен бути інтуїтивно зрозумілим та зручним у

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		21

використанні. Навігаційні елементи та компоненти інтерактивної взаємодії повинні бути виконані у зручній для користувача формі. Введення та виведення даних в систему, введення команд та результат їх виконання повинні виконуватися в інтерактивному режимі.

На інтерфейси користувача накладаються наступні вимоги:

- мінімальний час відгуку системи;
- повідомлення про критичні ситуації системи;
- отримання відповіді на будь-яку дію користувача;
- відображення процесу завантаження для ресурсомістких завдань.

2.2. Аналіз вимог до програмного продукту

Розглянемо високорівневі вимоги, яким повинен володіти онлайн сервіс замовлення та доставки квітів, щоб мати можливість безперервно виконувати покладені на нього функції, тим самим відповідно задовольнивши всі зазначені стандарти, специфікації та інші формальні документи. Зазначені умови, яких повинен дотримуватись користувач для роботи з даною інформаційною системою та вказані основні ділові переваги. Приведені можливості системи, обмеження та показники якості.

Бізнес-вимоги:

1. Основні цілі: проект створюється з метою надання користувачам доступу до інформації про систему замовлення та доставки квітів.
2. Представлення проекту: проект буде реалізовано у вигляді сайту та серверу, що містить структуровані ресурси з теми пов'язаними за даною тематикою.

Вимоги користувачів:

1. Зовнішні користувачі – авторизований користувач
 - Реєстрація за допомогою email
 - Можливість редагувати персональну інформацію

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		22

- Відправка відгуків про доставку квітів
- Перегляд останніх новин та акцій
- Перегляд на карті точок продажу квітів
- Можливість створення відгуків.

2. Характеристика об'єкту комп'ютеризації:

Користувач, після авторизації або реєстрації, буде мати змогу редагувати персональну інформацію, переглядати новини та акції квіткового магазину.

Функціональні вимоги:

- 1) Авторизація користувачів в системі: В системі повинна бути представлена можливість реєстрації користувача та присвоєння йому відповідної ролі.
- 2) Можливість збереження інформації: Система повинна зберігати інформацію і надавати можливість користувачу керувати нею.

Нефункціональні вимоги:

1. Сприйняття

- Час, потрібний для навчання інструментами роботи з інформаційною системою для звичайних користувачів – 10-15 хвилин, для досвідчених – 5 хвилин;
- Час відповіді системи для звичайних запитів не повинен перевищувати 5 секунд, а для більш складних запитів – 10 сек.
- Інтерфейс повинен бути інтуїтивно зручним для користувача та не вимагати додаткової підготовки.

2. Продуктивність

- Система повинна підтримувати мінімум 100 користувачів, що одночасно працюють, пов'язаних з спільною базою даних.

3. Можливість експлуатації

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		23

- Масштабування – система повинна мати можливість збільшувати потужності (продуктивність), зі збільшенням користувачів таким чином, щоб це аж ні як негативно не відобразилося на її роботі.

Системні вимоги:

1. Вимоги до середовища виконання:

Для підтримки високих стандартів безпеки у клієнта потрібно виключити деякі застарілі браузерери:

- версії Google Chrome нижче 6 версії;
- версії Internet Explorer нижче 7 версії;
- версії Firefox нижче 2.0;
- версії Safari нижче 2.1;
- всі версії Internet Explorer на Windows XP.

Мобільні браузерери, які не підтримуються сертифікатами SSL:

- Safari iOS нижче 4.0;
- Android нижче 3.0 (Honeycomb) ;
- Браузери Windows Phone версії нижче 7.

Антивимоги:

1. Не використовує інформацію, заборонену нормами моралі та законодавством.
2. Не використовує інформацію, що є недійсною або може зашкодити психіці користувачів.
3. Не містить реклами, принаймні тої, що не стосується тематики сайту.

Підсумовуючи, побудуємо діаграму варіантів використання (Use Case Diagram).

Діаграма варіантів використання у мові моделювання UML — це інструмент, який дозволяє представити функціональність системи через взаємодію акторів (організацій або користувачів) та конкретні сценарії використання, що називаються варіантами. Варіанти використання визначають набір дій, які система повинна виконувати для задоволення потреб акторів. Ці

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		24

діаграми є корисним інструментом для візуалізації функціональних вимог системи, що полегшує вибір пріоритетів та розробку дизайну [29].

Діаграма варіантів використання в нотатії UML представлена на рисунку 2.1.

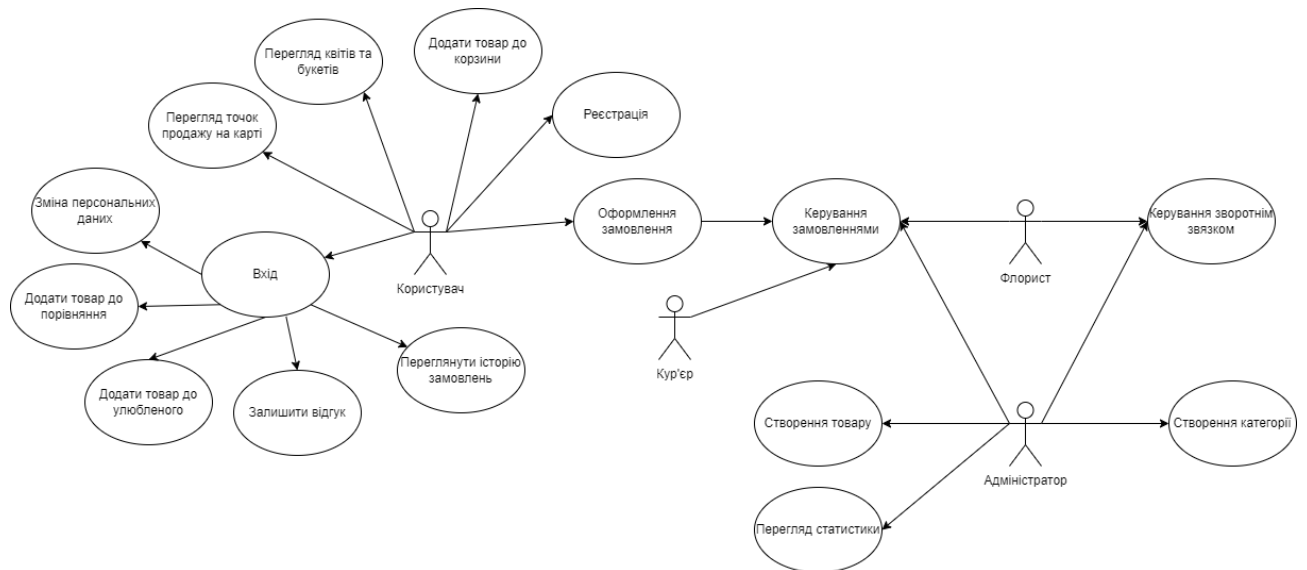


Рисунок 2.1 – Діаграма варіантів використання

Нижче наведено опис діаграми використання онлайн сервісу доставки квітів з участю користувача, адміністратора, флориста та кур'єра.

Гість, як користувач системи без облікового запису, має наступні можливості:

1. Вхід в систему або реєстрація нового облікового запису – гість може увійти до системи, використовуючи існуючі облікові дані, або зареєструвати новий обліковий запис, щоб мати можливість здійснювати покупки та користуватися додатковими функціями.
2. Перегляд точок продажу на карті – гість може переглядати мапу з розташуванням точок продажу квітів або квіткових магазинів.
3. Перегляд каталогу квітів – гість може переглядати доступні на платформі квіти та квіткові композиції і обирати серед них ті, які йому подобаються.
4. Перегляд новинок – гість може отримувати актуальну інформацію щодо нових пропозицій, акцій, або подій, пов'язаних з послугами платформи.

5. Додавання товару до кошика – гість може вибирати продукти з каталогу та додавати їх до своєї кошика для подальшого оформлення замовлення. Авторизований користувач, який має обліковий запис на платформі, додатково до можливостей гостя, може виконувати такі дії:

1. Редагування персональних даних – користувач має можливість змінювати свої особисті дані, такі як адреса доставки, контактна інформація тощо.
2. Залишення відгуку – користувач може залишати відгуки про куплені товари або процес обслуговування, що допомагає іншим користувачам приймати рішення щодо покупки.
3. Перегляд історії замовлень – користувач має доступ до історії своїх попередніх замовлень, включаючи деталі про товари, дати і статуси замовлень. Це дозволяє користувачеві відстежувати свої покупки та контролювати їх стан.

Адміністратор, може виконувати такі дії:

1. Додавання товарів – адміністратор може додавати нові товари до каталогу, забезпечуючи користувачів актуальними пропозиціями.
2. Редагування товарів – адміністратор має можливість оновлювати інформацію про існуючі товари, включаючи опис, ціну та зображення.
3. Видалення товарів – адміністратор може видаляти товари з каталогу, якщо вони більше не доступні або неактуальні.
4. Додавання товарів – адміністратор може додавати нові товари до каталогу, забезпечуючи користувачів актуальними пропозиціями.
5. Редагування товарів – адміністратор має можливість оновлювати інформацію про існуючі товари, включаючи опис, ціну та зображення.
6. Видалення товарів – адміністратор може видаляти товари з каталогу, якщо вони більше не доступні або неактуальні.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		26

7. Перегляд статистики – адміністратор має доступ до статистичних даних щодо продажів, популярності товарів та інших ключових показників ефективності сервісу.
8. Перегляд зворотного зв'язку – адміністратор може переглядати відгуки та коментарі користувачів, щоб поліпшити якість обслуговування та продукції.

Флорист, може виконувати такі дії:

1. Збір замовлення – флорист відповідає за збирання замовлень, які надійшли через платформу. Він збирає необхідні квіти та komponує їх у букети відповідно до вимог та побажань клієнтів.
2. Редагування або видалення наявних букетів на сайті – флорист може редагувати існуючі букети на сайті, змінюючи їх опис, фотографії або ціни. Він також може видаляти букети, якщо вони більше не доступні або відповідають стандартам платформи.
3. Додавання квітів та букетів – флорист має можливість додавати нові види квітів та букетів на платформу. Він може завантажувати фотографії, додавати опис та встановлювати ціни для нових продуктів.
4. Перегляд замовлень – флорист може переглядати список замовлень, які надійшли на обробку. Це дозволяє флористу відстежувати, які замовлення вимагають його уваги та обробки.
5. Редагування статусу замовлення – флорист може змінювати статус замовлення, вказуючи, чи він оброблений, в процесі виконання, чекає на доставку або вже доставлений. Це допомагає відстежувати процес виконання замовлень та надавати користувачам актуальну інформацію про їх статус.

Кур'єр, може виконувати такі дії:

1. Отримання інформації про замовлення – кур'єр отримує доступ до інформації про замовлення, які він повинен доставити. Це може включати

деталі про товари, адресу доставки, контактні дані отримувача та будь-які спеціальні інструкції щодо доставки.

2. Підтвердження отримання замовлення та отримання необхідних відомостей – кур'єр підтверджує отримання замовлення та забирає необхідні відомості для доставки, такі як точна адреса та контактні дані отримувача. Це допомагає кур'єру ефективно планувати маршрут доставки та забезпечити успішне виконання замовлення.
3. Позначення замовлення як "доставлено" та вказання часу доставки – після успішної доставки кур'єр позначає замовлення як "доставлено" у системі та може вказати час, коли замовлення було передано отримувачу. Це дозволяє платформі та клієнтам відстежувати стан доставки та підтверджує успішне завершення процесу доставки.

Діаграма використання допомагає узагальнити та візуалізувати взаємодію між усіма учасниками системи та процеси, які вони виконують.

2.3. Алгоритм роботи та стани програмної системи

Діаграма діяльності є ключовою графічною представленням для опису динамічних аспектів системи. В сутності, це вдосконалена форма блок-схеми, яка моделює потік виконання від однієї дії до іншої [4].

Діаграма діяльності процесу виконання замовлення для онлайн сервісу доставки квітів включає три основних учасника: клієнта, флориста та кур'єра. Діаграма розпочинається, коли користувач вибирає та підтверджує замовлення після оформлення. Флорист починає обробку замовлення, де перевіряє чи товар наявності, якщо ні – дія закінчується, якщо так – відбувається збір замовлення, після чого йде відгрузка замовлення, де замовлення приймає вже кур'єр. Кур'єр приймає замовлення, після чого відбувається доставка замовлення, при прибутті кур'єра на місце, кур'єр підтверджує доставку замовлення і доставка вважається завершеною.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		28

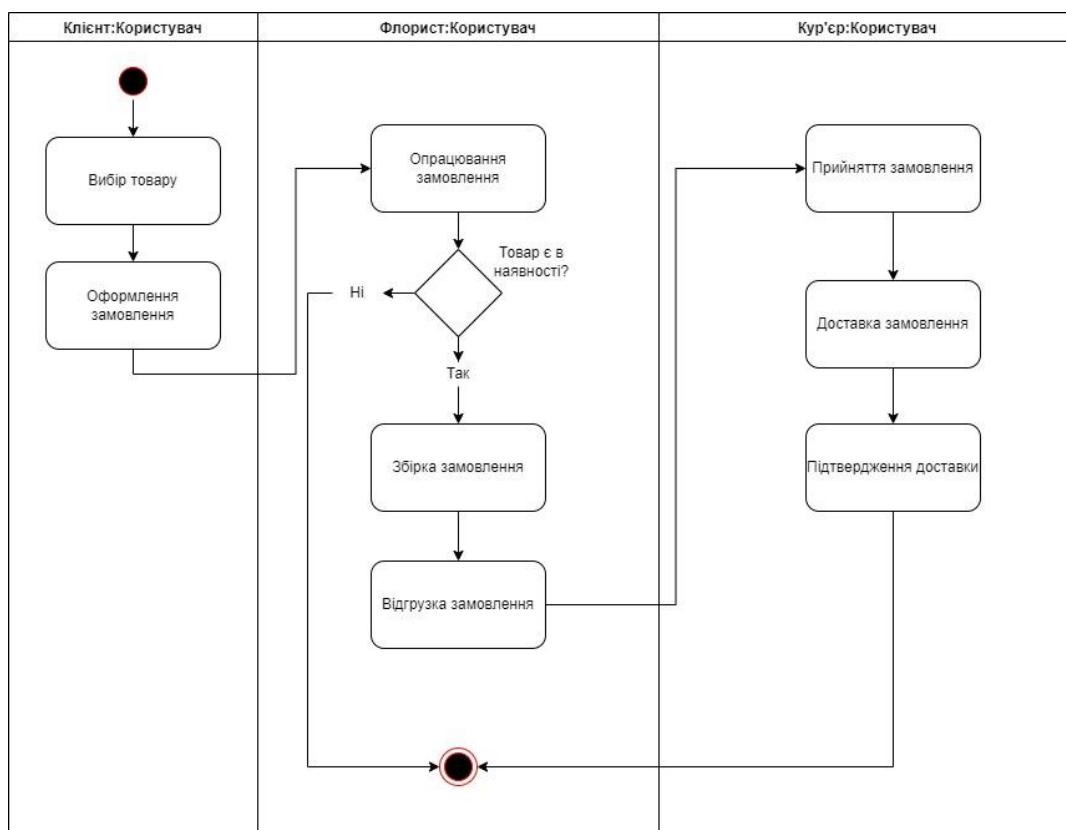


Рисунок 2.2 – Діаграма діяльності виконання замовлення

Діаграма діяльності процесу оформлення замовлення для онлайн сервісу доставки квітів включає два основних учасника: користувача та систему. Користувач заходить у віртуальну корзину на сайті, щоб переглянути вибрані товари. Система відображає вміст корзини з усіма доданими товарами та загальною сумою. Користувач може обрати одну з наступних опцій: видалити товар з корзини або оформити замовлення.

Якщо користувач обирає опцію "Видалити товар", система видаляє товар з корзини та перераховує загальну суму замовлення. Якщо користувач обирає опцію "Оформити замовлення", система відображає форму для оформлення замовлення. Користувач вводить свої контактні дані (ім'я, номер телефону, електронну пошту), адресу доставки та обирає спосіб доставки з доступних варіантів. Після введення даних користувач підтверджує оформлення замовлення.

Система перевіряє коректність введених даних. Якщо дані коректні, замовлення відправляється на виконання. Якщо дані некоректні, система

відображає повідомлення про помилку введених даних та просить виправити помилки.

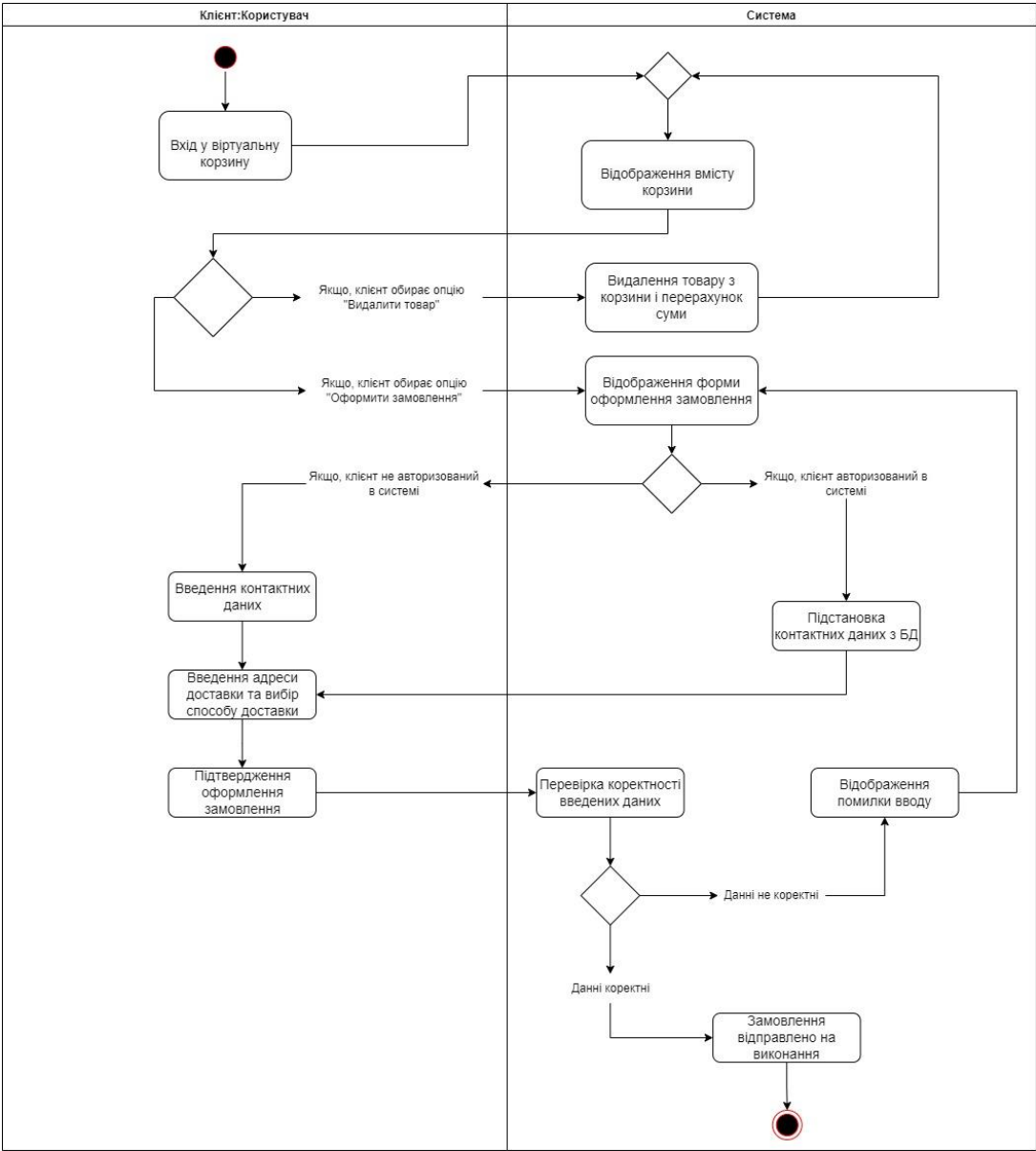


Рисунок 2.3 – Діаграма діяльності оформлення замовлення

2.4. Об'єктно-орієнтована модель системи

Об'єктно-орієнтовану модель системи прийнято розглядати у вигляді діаграми класів (class diagram). Для побудови даної діаграми спочатку потрібно розібратися, що воно собою представляє. Діаграма класів – це тип діаграми

статичної структури, який описує структуру системи, показуючи класи системи, їхні атрибути, операції (або методи) і зв'язки між об'єктами [30].

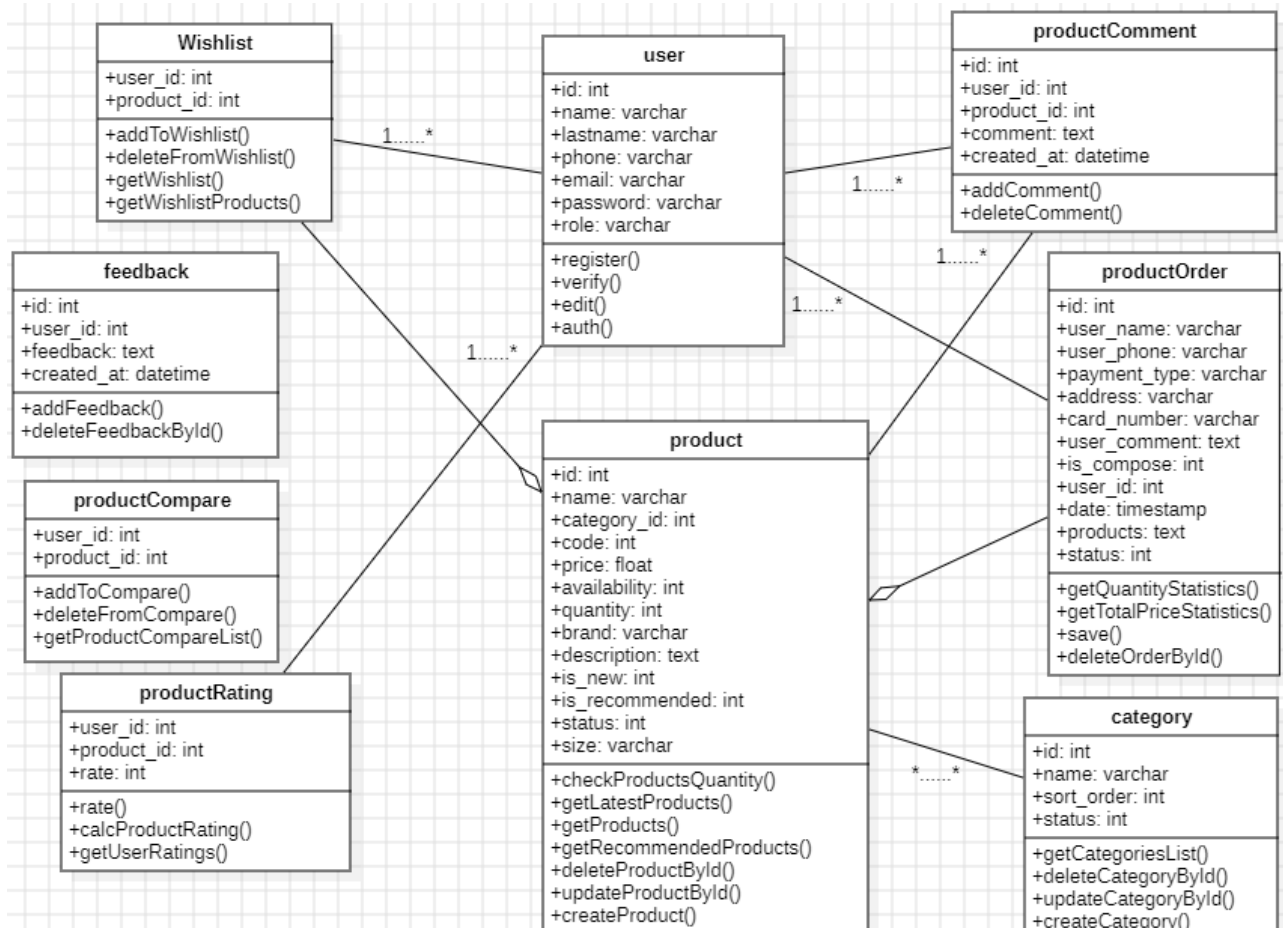


Рисунок 2.4 – Діаграма класів

Нижче наведено опис класів, а також їх атрибутів та методів:

1. User (Користувач)

- Опис: Цей клас описує користувача системи.
- Атрибути:
 - id: Унікальний ідентифікатор користувача.
 - user_name: Ім'я користувача.
 - last_name: Прізвище користувача.
 - phone: Номер телефону користувача.
 - email: Адреса електронної пошти користувача.
 - password: Пароль користувача.
 - role: Роль користувача (наприклад, адміністратор, користувач).

- **Методи:**
 - register(): Зареєструвати нового користувача.
 - verify(): Перевірити адресу електронної пошти користувача.
 - edit(): Редагувати профіль користувача.
 - auth(): Авторизуватися в системі.

2. Product (Продукт)

- **Опис:** Цей клас описує продукт, який продається в системі.
- **Атрибути:**
 - id: Унікальний ідентифікатор продукту.
 - name: Назва продукту.
 - category_id: Ідентифікатор категорії, до якої належить продукт.
 - code: Код продукту.
 - price: Ціна продукту.
 - availability: Доступність продукту (наприклад, "в наявності", "немає в наявності").
 - quantity: Кількість доступних екземплярів продукту.
 - brand: Бренд продукту.
 - description: Опис продукту.
 - status: Статус продукту (наприклад, "активний", "неактивний").
- **Методи:**
 - save(): Зберегти продукт.
 - deleteOrderByld(): Видалити замовлення за ID.
 - updateProductById(): Оновити продукт за ID.
 - createProduct(): Створити новий продукт.

3. Category (Категорія)

- **Опис:** Цей клас описує категорію продуктів.
- **Атрибути:**
 - id: Унікальний ідентифікатор категорії.
 - name: Назва категорії.

- sort_order: Порядок сортування категорій.
- status: Статус категорії (наприклад, "активна", "неактивна").
- Методи:
 - deleteCategoryById(): Видалити категорію за ID.
 - updateCategoryById(): Оновити категорію за ID.
 - createCategory(): Створити нову категорію.

4. ProductOrder (Замовлення продукту)

- Опис: Цей клас описує замовлення продукту.
- Атрибути:
 - id: Унікальний ідентифікатор замовлення.
 - user_id: Ідентифікатор користувача, який зробив замовлення.
 - user_name: Ім'я користувача, який зробив замовлення.
 - user_phone: Номер телефону користувача, який зробив замовлення.
 - payment_type: Тип оплати замовлення (наприклад, "готівка", "карта").
 - address: Адреса доставки замовлення.
 - card_number: Номер картки, яка використовується для оплати замовлення.
- Методи:
 - addFeedback(): Додати відгук до замовлення.
 - deleteFeedbackById(): Видалити відгук з замовлення за ID.

5. ProductComment (Коментар до продукту)

- Опис: Цей клас описує коментар до продукту.
- Атрибути:
 - id: Унікальний ідентифікатор коментаря.
 - user_id: Ідентифікатор користувача, який залишив коментар.
 - product_id: Ідентифікатор продукту, до якого стосується коментар.

- comment: Текст коментаря.
- rating: Рейтинг продукту, даний користувачем (від 1 до 5 зірок).
- comment_date: Дата та час залишення коментаря.
- Методи:
 - addComment(product_id, user_id, comment, rating): Додати коментар до продукту.
 - getComments(product_id): Отримати всі коментарі до продукту.
 - getCommentById(comment_id): Отримати коментар за його ID.
 - updateComment(comment_id, new_comment, new_rating): Оновити коментар.
 - deleteComment(comment_id): Видалити коментар.

6. Wishlist (Список бажань)

- Опис: Цей клас описує список бажань користувача.
- Атрибути:
 - user_id: Ідентифікатор користувача
 - product_id: Ідентифікатор продукту
- Методи:
 - addToWishlist(): Додає продукт до списку бажань
 - deleteFromWishlist(): Видаляє продукт із списку бажань
 - getWishlist(): Отримує список бажань користувача
 - getWishlistProducts(): Отримує список продуктів у списку бажань

7. Feedback (Зворотний зв'язок)

- Опис: Цей клас описує повідомлення від користувачів.
- Атрибути:
 - id: Ідентифікатор відгуку
 - user_id: Ідентифікатор користувача
 - product_id: Ідентифікатор продукту
 - feedback: Текст відгуку
 - created_at: Дата та час створення відгуку

- Методи:
 - `addFeedback()`: Додає відгук до продукту
 - `deleteFeedbackById()`: Видаляє відгук за його ідентифікатором

8. Product Compare (Порівняння продуктів)

- Опис: Цей клас описує порівняння продуктів.
- Атрибути:
 - `user_id`: Ідентифікатор користувача
 - `product_id`: Ідентифікатор продукту
- Методи:
 - `addToCompare()`: Додає продукт до списку для порівняння
 - `deleteFromCompare()`: Видаляє продукт із списку для порівняння
 - `getProductCompareList()`: Отримує список продуктів для порівняння користувача

9. Product Rating (Рейтинг продукту)

- Опис: Цей клас описує рейтинг продукту.
- Атрибути:
 - `user_id`: Ідентифікатор користувача
 - `product_id`: Ідентифікатор продукту
 - `rate`: Рейтинг користувача продукту
- Методи:
 - `rate()`: Оцінює продукт
 - `calcProductRating()`: Розраховує середню оцінку продукту
 - `getUserRatings()`: Отримує рейтинги продуктів користувача

2.5. Взаємодія об'єктів системи

Для візуалізації взаємодії об'єктів у системі, в UML зазвичай використовують діаграми послідовності.

Ці діаграми детально описують порядок виконання операцій та фіксують взаємодію об'єктів під час співпраці. Зокрема, вони концентруються на часі, використовуючи вертикальну ось для відображення порядку взаємодії та моменти відправлення повідомлень [29].

З метою продемонстрування процесу обробки конкретного замовлення вирішено створити діаграму послідовності для модуля "Обрати товар" та "Зробити замовлення".

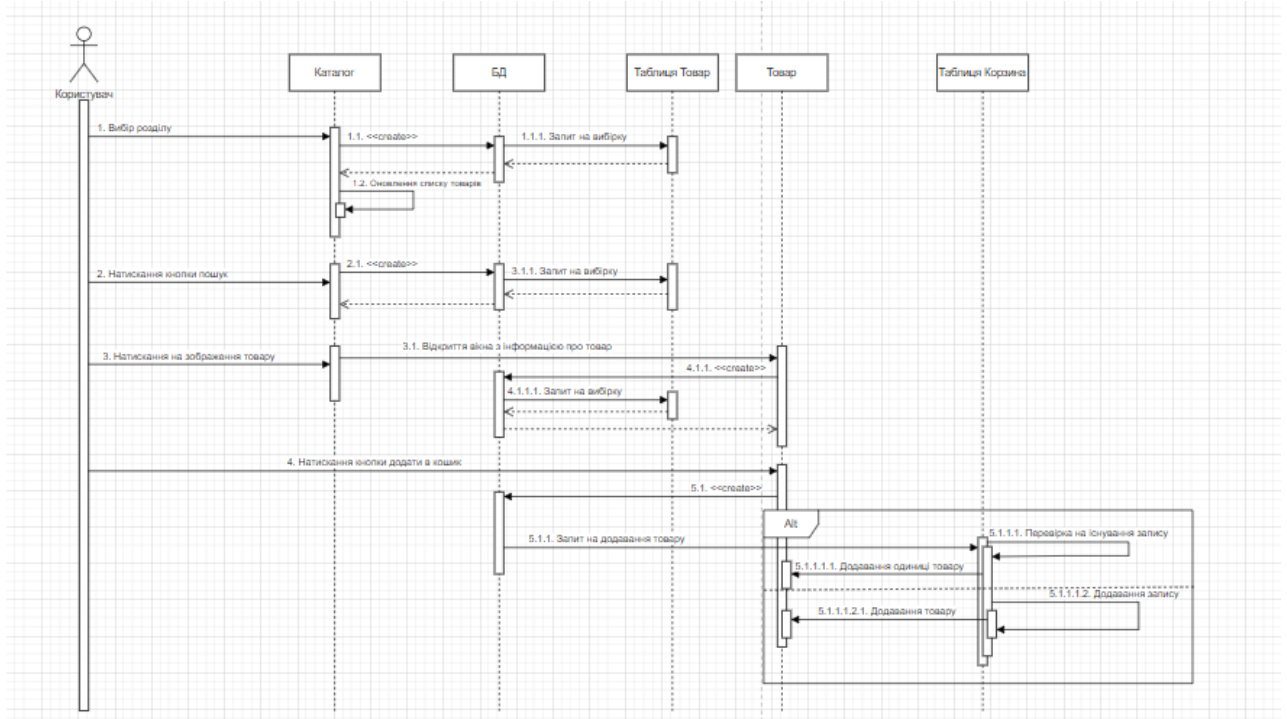


Рисунок 2.5 – Діаграма послідовності для модуля вибору товару

Користувач взаємодіє з інтерфейсом магазину та обирає товар для покупки. Браузер відправляє запит на сервер для отримання інформації про товари. Веб-сервер обробляє запит і взаємодіє з базою даних для отримання інформації про товари. Витягується інформація про товари з таблиці товарів в базі даних. Веб-сервер відправляє клієнту список товарів для відображення на сторінці. Користувач обирає конкретний товар та натискає кнопку "Додати в корзину". JavaScript у браузері генерує подію "Додавання товару в корзину" і відправляє її на сервер. Веб-сервер отримує подію та взаємодіє з базою даних для оновлення вмісту корзини. Додається запис про вибраний товар до таблиці

корзини. Веб-сервер відправляє підтвердження користувачеві. Користувач може продовжити вибір товарів або переглядати вміст своєї корзини.

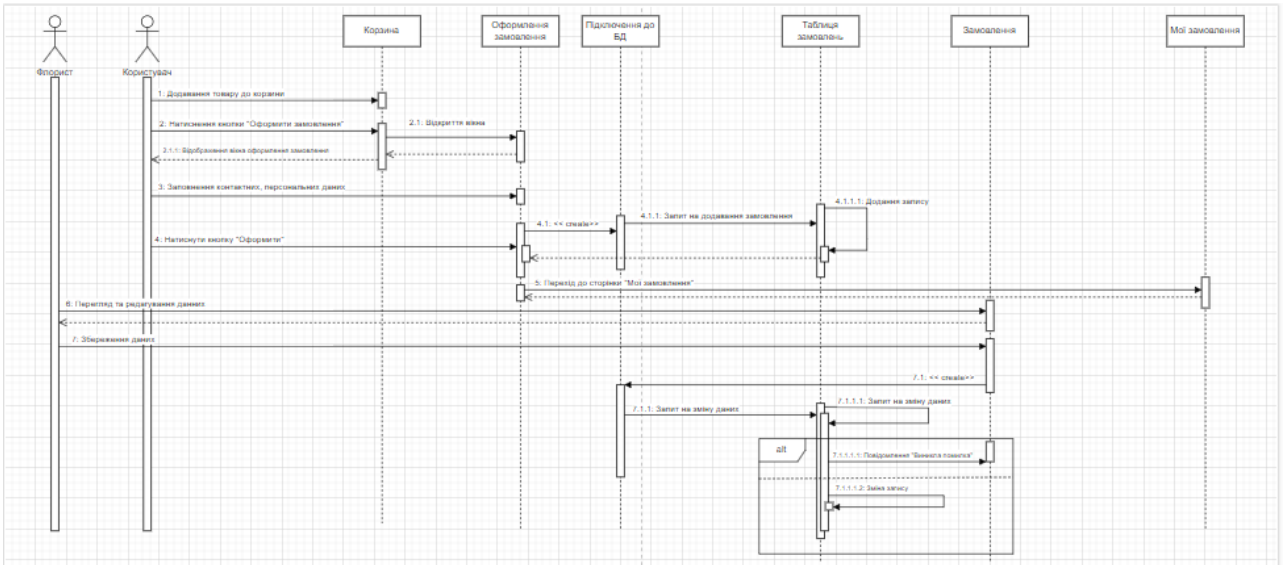


Рисунок 2.6 – Діаграма послідовності для модуля оформлення замовлення

Клієнт обирає товари, додає їх до кошика, переходить до оформлення замовлення та вводить свої дані для доставки і платіжну інформацію. Інтернет-магазин перевіряє наявність товарів, надсилає інформацію до системи обробки платежів, яка обробляє платіж і повідомляє результат. Потім інтернет-магазин передає інформацію до служби доставки, яка планує та виконує доставку, повідомляючи клієнта. Клієнт підтверджує отримання замовлення.

2.6. Розробка бази даних системи

Реалізована база даних MySQL складається з 9 таблиць, які містять у собі всі дані для роботи програми.

Структура бази даних наведена на Рисунок 2.7.

Таблиця «Product» призначена для зберігання інформації про продукти.

Таблиця 2.2

Опис полів таблиці «Product»

Назва	Тип даних	ПК	ЗК	Опис поля
id	int	+	—	ID продукту
name	varchar	—	—	Назва продукту
category_id	int	—	+	Ідентифікатор категорії
code	int	—	—	Код букету
price	int	—	—	Ціна букету
availability	int	—	—	Наявність букету (1 - так, 0 - ні)
quantity	int	—	—	Кількість
brand	varchar	—	—	Країна походження
description	text	—	—	Опис продукту
is_new	int	—	—	Новий продукт (1 - так, 0 - ні)
is_recommended	int	—	—	Рекомендований продукт (1 - так, 0 - ні)
status	int	—	—	Статус продукту (1 - активний, 0 - не активний)
size	varchar	—	—	Розмір

Таблиця «Category» призначена для зберігання категорій продуктів.

Таблиця 2.3

Опис полів таблиці «Category»

Назва	Тип даних	ПК	ЗК	Опис поля
id	int	+	—	ID категорії
name	varchar	—	—	Назва продукту
sort_order	int	—	—	Порядок сортування
status	int	—	—	Статус категорії (1 - активна, 0 - не активна)

Таблиця «Feedback» призначена для зберігання відгуків користувачів.

Таблиця 2.4

Опис полів таблиці «Feedback»

Назва	Тип даних	ПК	ЗК	Опис поля
id	int	+	—	ID відгуку
user_id	int	—	+	Ідентифікатор користувача
feedback	text	—	—	Текст відгуку
created_at	datetime	—	—	Дата створення відгуку

Таблиця «Wishlist» призначена для зберігання списків бажань користувачів.

Таблиця 2.5

Опис полів таблиці «Wishlist»

Назва	Тип даних	ПК	ЗК	Опис поля
user_id	int	+	+	Ідентифікатор користувача
product_id	int	+	+	Ідентифікатор продукту

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		40

Таблиця «ProductComment» призначена для зберігання коментарів до продуктів.

Таблиця 2.6

Опис полів таблиці «ProductComment»

Назва	Тип даних	ПК	ЗК	Опис поля
id	int	+	–	ID коментаря
user_id	int	–	+	Ідентифікатор користувача
product_id	int	–	+	Ідентифікатор продукту
comment	text	–	–	Текст коментаря
created_at	datetime	–	–	Дата створення коментаря

Таблиця «ProductOrder» призначена для зберігання інформації про замовлення продуктів.

Таблиця 2.7

Опис полів таблиці «ProductOrder»

Назва	Тип даних	ПК	ЗК	Опис поля
1.	2.	3.	4.	5.
id	int	+	–	ID замовлення
user_name	varchar	–	+	Ім'я користувача
user_phone	varchar	–	+	Номер телефону користувача
payment_type	varchar	–	–	Тип оплати
address	varchar	–	–	Адреса доставки
card_number	varchar	–	–	Номер картки
user_comment	text	–	–	Коментар користувача
is_composite	int	–	–	Композитне замовлення (1 - так, 0 - ні)
user_id	int	–	+	Ідентифікатор користувача

Продовження таблиці 2.7

1.	2.	3.	4.	5.
date	timestamp	—	—	Дата замовлення
products	text	—	—	Перелік букетів
status	int	—	—	Статус замовлення (1 - активне, 0 - не активне)

Таблиця «ProductCompare» призначена для зберігання інформації про продукти, які користувачі порівнюють.

Таблиця 2.8

Опис полів таблиці «ProductCompare»

Назва	Тип даних	ПК	ЗК	Опис поля
user_id	int	+	+	Ідентифікатор користувача
product_id	int	+	+	Ідентифікатор продукту

Таблиця «ProductRating» призначена для зберігання рейтингів продуктів від користувачів.

Таблиця 2.9

Опис полів таблиці «ProductRating»

Назва	Тип даних	ПК	ЗК	Опис поля
user_id	int	+	+	Ідентифікатор користувача
product_id	int	+	+	Ідентифікатор продукту
rate	int	—	—	Рейтинг продукту

Висновки до другого розділу

У процесі розробки було виконано комплекс завдань, що спрямовані на створення ефективної програмної системи.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		42

На першому етапі, шляхом аналізу технічного завдання, були обрані необхідні засоби моделювання, а також проведено докладний аналіз вимог до програмного продукту. Цей етап визначив основні параметри та характеристики системи.

Далі, наступний крок передбачав створення логічної моделі програмної системи. Це включало в себе створення діаграми діяльності, яка ілюструє алгоритми роботи та можливі стани програмної системи, а також діаграми класів для представлення об'єктно-орієнтованої моделі системи. Це сприяло чіткому визначенню внутрішньої структури системи та її функціональних взаємозв'язків.

У результаті виконаних етапів було досягнуто глибокого розуміння функціоналу системи, забезпечено її ефективність та гнучкість. Застосування вищезгаданих методів та підходів сприяло створенню програмного продукту, який відповідає визначеним вимогам та готовий до подальшої реалізації та використання.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		43

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ

3.1. Порядок встановлення та налаштування системи

Вибір архітектури та моделі взаємодії є критично важливим етапом у розробці веб-додатків, що одночасно є одним із найскладніших завдань. Архітектура веб-додатків визначає схему взаємодії між їх компонентами, а спосіб цієї взаємодії визначає майбутню стійкість, ефективність та безпеку самого додатка [28].

Онлайн-сервісу замовлення та доставки квітів використовує шаблон архітектури MVC (Модель-Вид-Контролер) (рис. 3.1), де модель відповідає за дані та бізнес-логіку, вид представляє інформацію для користувача, а контролер керує логікою додатку [12].

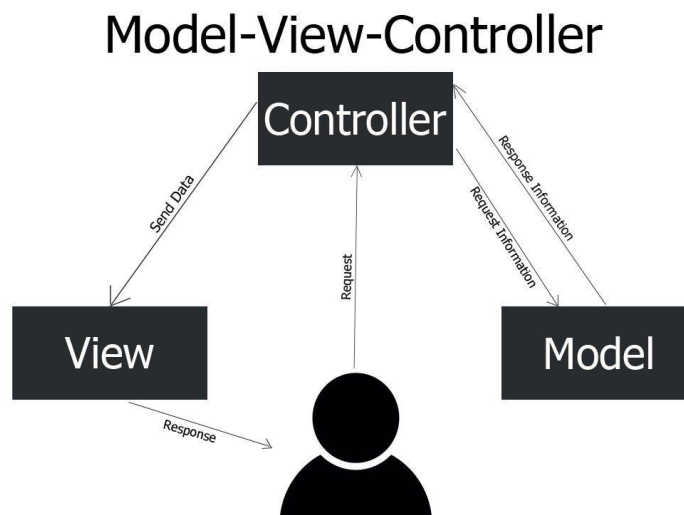


Рисунок 3.1 – Модель шаблону проектування MVC

Контролер призначений для обробки запитів користувача, які надходять у вигляді HTTP-запитів GET або POST під час взаємодії з інтерфейсом. Його головною функцією є виклик і координування необхідних ресурсів та об'єктів для виконання дій. Зазвичай контролер взаємодіє з моделлю для отримання необхідних даних та вибирає відповідний вид для відображення. Модель забезпечує контролер необхідною інформацією з бази даних. Отримані дані

передаються від моделі до виду, який відповідає за представлення даних у різних форматах. Веб-додаток зазвичай складається з набору контролерів, моделей та видів.

Однією з основних переваг використання шаблону MVC є чітке розділення логіки представлення (інтерфейсу користувача) та логіки додатку. Наприклад, проблема підтримки різних типів пристроїв стає важливою у сучасному світі. Відображення вмісту веб-сторінки повинно відрізнятися в залежності від типу пристрою, який використовується користувачем. У таких випадках модель може повертати однакові дані, але контролер обирає різні представлення для відображення інформації [12].

Структура онлайн-сервісу була створена самостійно і має наступний вигляд (рис. 3.2).

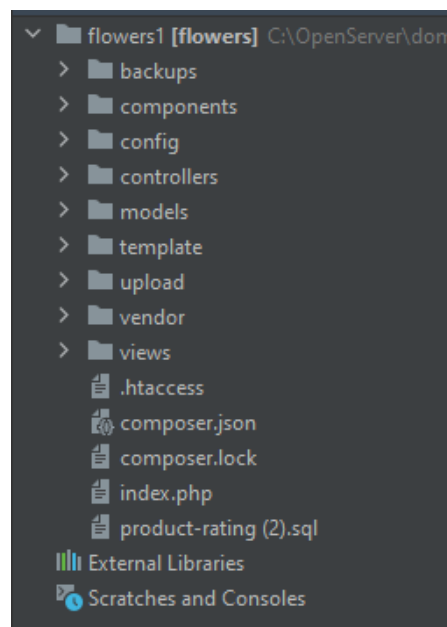


Рисунок 3.2 – Файлова структура онлайн-сервісу

Онлайн-сервісу з назвою "flowers1" має кореневу папку з такою ж назвою. У папці "components" розташований клас "Db", в якому є функція getConnection(), призначена для забезпечення з'єднання з базою даних. В папці "config" розміщені основні налаштування, представлені в звичайних PHP-файлах у вигляді масивів ключів та значень, необхідних для коректної роботи всього онлайн-сервісу. Наприклад, у файлі "db_params" визначені основні параметри

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		45

для підключення до бази даних, такі як "host", "dbname", "user", "password". Підхід, при якому основні налаштування зберігаються у одному файлі, є поширеним, оскільки це спрощує внесення змін у майбутньому, наприклад, у параметрах підключення до бази даних, що запобігає виникненню помилок під час розробки додатку. Ще один файл у папці "config" - "routes", містить інформацію, необхідну для коректного відображення вмісту кожної зі сторінок онлайн-сервісу. В папці "controllers" розміщені контролери, такі як ProductController або CatalogController, які використовуються у проекті.

Класи, що містять методи для взаємодії з базою даних або іншими сервісами, зберігаються у папці "models". Стили, які використовуються на сторінках онлайн-сервісу, описані у файлах папки "template". Кожній сторінці відповідає певний шаблон з необхідним вмістом, а файли з видами розташовані у папці "views".

Під час проектування структури онлайн-сервісу було вирішено, що уся інформація буде зберігатися у базі даних. В якості системи управління базами даних обрано MySQL. Усі необхідні налаштування для коректної взаємодії з базою даних та інтернет-ресурсом прописані у файлі "db_params.php" (рис. 3.3).

```
<?php

return array(
    'host' => '127.0.0.1',
    'dbname' => 'mvs_site',
    'user' => 'root',
    'password' => '',
);
```

Рисунок 3.3 – Підключення до баз даних

3.2. Розробка основних функцій магазину

На рисунку 3.3 зображено фрагмент коду реалізації методу авторизації, де функція actionLogin() призначена для авторизації користувача на сайті. Починається з отримання списку категорій за допомогою методу

getCategoriesList() класу Category. Перевіряє, чи були надіслані дані форми з даними для входу. Якщо дані були надіслані, функція отримує значення email та пароль з POST-запиту.

Проводиться перевірка валідності email за допомогою методу checkEmail() класу User. Якщо email не відповідає вимогам, додається відповідне повідомлення про помилку у масив помилок. Викликається метод checkUserData(), який перевіряє, чи відповідають введені дані даним в базі даних. Якщо дані неправильні або користувач не існує, додається відповідне повідомлення про помилку у масив помилок.

Якщо користувач існує та введені дані вірні, перевіряється, чи обліковий запис користувача верифікований (поле is_verified дорівнює 1). Якщо обліковий запис не верифікований, додається відповідне повідомлення про помилку у масив помилок.

Якщо ж всі перевірки пройдено успішно, викликається метод auth(), який здійснює авторизацію користувача, та відбувається перенаправлення користувача на його особистий кабінет за допомогою заголовку Location. У кінці функція викликає шаблон сторінки входу для користувача та повертає true.

```
public function actionLogin()
{
    $categories = Category::getCategoriesList();
    $email = false;
    $password = false;
    if (isset($_POST['submit'])) {
        $email = $_POST['email'];
        $password = $_POST['password'];
        $errors = array();
        if (!User::checkEmail($email)) {
            $errors[] = 'Невірний email';
        }
        $user = User::checkUserData($email, $password);
        if ($user == false) {
            $errors[] = 'Неправильні дані для входу на сайт';
        } else if ($user['is_verified'] == '0') {
            $errors[] = 'Обліковий запис не верифікований';
        } else {
            User::auth($user['id']);
            header("Location: /cabinet/");
        }
    }
    require_once(ROOT . '/views/user/login.php');
    return true;
}
```

Рисунок 3.4 – Функція авторизації

Метод реєстрації аналогічно до попередньої функції, де функція `actionRegister()` відповідає за реєстрацію нового користувача на сайті. Починається з отримання з'єднання з базою даних за допомогою методу `getConnection()` класу `Db`. Отримує список категорій за допомогою методу `getCategoriesList()` класу `Category`. Ініціалізує змінні для зберігання даних, що надходять з форми реєстрації. Перевіряє, чи була надіслана форма з даними для реєстрації.

Якщо дані були надіслані, проводиться перевірка валідності введених даних за допомогою методів класу `User`, таких як `checkName()`, `checklastname()`, `checkPhone()`, `checkEmail()` і `checkPassword()`. Також перевіряється, чи вже існує користувач з введеним email або телефонним номером.

Якщо дані введені коректно та унікально, викликається метод `register()` класу `User`, який додає нового користувача в базу даних. Після цього викликається метод `verifyUser()`, який відправляє електронний лист з посиланням для підтвердження облікового запису. Функція завершується поверненням значення `true`, щоб підтвердити успішне виконання.

```
public function actionRegister()
{
    $db = Db::getConnection();

    $categories = Category::getCategoriesList();

    $name = '';
    $email = '';
    $password = '';
    $lastname = '';
    $phone = '';

    $result = false;

    if (isset($_POST['submit'])) {
        $name = $_POST['name'];
        $password = $_POST['password'];
        $lastname = $_POST['lastname'];
        $phone = $_POST['phone'];
        $email = $_POST['email'];

        $errors = array();

        if (!User::checkName($name)) {
            $errors[] = "Ім'я не повинно бути коротшим за 4 символи";
        }
    }
}
```

Рисунок 3.5 – Функція реєстрації

```

        if (!User::checkLastname($lastname)) {
            $errors[] = 'Прізвище не повинно бути коротшим за 2 символи';
        }
        if (!User::checkPhone($phone)) {
            $errors[] = 'Телефон введено неправильно!
            Приклад: 097 9714 664';
        }
        if (!User::checkEmail($email)) {
            $errors[] = 'Неправильний email';
        }
        if (!User::checkPassword($password)) {
            $errors[] = 'Пароль не повинен бути коротшим за 8 символів';
        }
        if (User::checkEmailExists($email)) {
            $errors[] = 'Такий email вже використовується';
        }
        if (User::checkPhoneExists($phone)) {
            $errors[] = 'Такий телефон уже використовується';
        }
        if (count($errors) == 0) {
            $result = User::register($name, $lastname, $phone, $email, $password);
            self::verifyUser(User::getUserByEmail($email));
        }
    }
    require_once(ROOT . '/views/user/register.php');
    return true;
}

```

Рисунок 3.6 – Продовження функції реєстрації

Зображено фрагмент коду на рисунку 3.7 в якому описана функція додавання товару до корзини. Ця функція, `addProduct($id)`, відповідає за додавання продукту до кошика користувача.

```

public static function addProduct($id)
{
    $product = Product::getProductById($id);

    if (!self::canAddMore($product)) {
        return array(
            'count' => self::countItems(),
            'canAddMore' => false
        );
    }
    $id = intval($id);
    $productsInCart = array();
    if (isset($_SESSION['products'])) {
        $productsInCart = $_SESSION['products'];
    }
    if (array_key_exists($id, $productsInCart)) {
        $productsInCart[$id]++;
    } else {
        $productsInCart[$id] = 1;
    }
    $_SESSION['products'] = $productsInCart;
    return array(
        'count' => self::countItems(),
        'canAddMore' => self::canAddMore($product)
    );
}

```

Рисунок 3.7 – Функція додавання товару до корзини

Спочатку, вона отримує об'єкт продукту за його ідентифікатором за допомогою методу `getProductById($id)` класу `Product`. Перевіряє, чи можна додати ще один такий самий продукт до кошика за допомогою методу `canAddMore($product)`. Якщо неможливо, повертається масив з даними про кількість товарів у кошику та індикатором `canAddMore` в значенні `false`. Якщо можливо, продовжується операція. Ідентифікатор продукту перетворюється в ціле число.

Отримується поточний список продуктів у кошику. Якщо в кошику вже є продукт з таким ідентифікатором, збільшується його кількість на одиницю. В іншому випадку, продукт додається до кошика з кількістю одна. Оновлений список продуктів у кошику зберігається у сесії.

Повертається масив з даними про кількість товарів у кошику та індикатором `canAddMore`, який вказує, чи можна додати ще один продукт до кошика.

Для видалення товару з корзини реалізовано функцію `deleteProduct($id)` (рис. 3.8), відповідає за видалення продукту з кошика користувача за його ідентифікатором.

```
public static function deleteProduct($id)
{
    $productsInCart = self::getProducts();
    unset($productsInCart[$id]);
    $_SESSION['products'] = $productsInCart;
}
```

Рисунок 3.8 – Функція видалення товару з корзини

Спочатку отримується список продуктів у кошику за допомогою методу `getProducts()`. Потім за допомогою функції `unset()` видаляється продукт з вказаним ідентифікатором зі списку продуктів у кошику. Оновлений список продуктів у кошику зберігається у сесії.

На рисунку 3.9 зображено функцію додавання товару до списку побажань.

```
public static function addToWishlist($userId, $productId)
{
    $db = Db::getConnection();

    $sql = "INSERT INTO wishlist (user_id, product_id) VALUES(:userId, :productId) ON DUPLICATE
    KEY UPDATE user_id=user_id, product_id=product_id";

    $result = $db->prepare($sql);
    $result->bindParam(':userId', $userId, PDO::PARAM_INT);
    $result->bindParam(':productId', $productId, PDO::PARAM_INT);

    return $result->execute();
}
```

Рисунок 3.9 – Функція додавання товару до списку побажань

Спочатку вона встановлює з'єднання з базою даних. Потім вона формує SQL-запит для вставки запису в таблицю wishlist. Цей запит використовує конструкцію INSERT INTO ... ON DUPLICATE KEY UPDATE, що означає, що якщо у таблиці вже існує запис з такими ж ключами, як у новому записі, то буде виконане оновлення цього запису замість вставки нового.

Параметри userId і productId вставляються у відповідні місця запиту з використанням підготовлених виразів для запобігання SQL-ін'єкціям. Запит виконується за допомогою методу execute() об'єкта PDOStatement.

Результат виконання запиту повертається: true, якщо операція успішно виконана, або false, якщо сталася помилка.

Для порівняльної характеристики товару розроблено функцію, яка зображена на рисунку 3.10.

```
public static function addToCompare($userId, $productId)
{
    $db = Db::getConnection();

    $sql = "INSERT INTO product_compare (user_id, product_id) VALUES(:userId, :productId) ON DUPLICATE
    KEY UPDATE user_id=user_id, product_id=product_id";

    $result = $db->prepare($sql);
    $result->bindParam(':userId', $userId, PDO::PARAM_INT);
    $result->bindParam(':productId', $productId, PDO::PARAM_INT);

    return $result->execute();
}
```

Рисунок 3.10 – Функція додавання товару порівняння

Спочатку встановлюється з'єднання з базою даних. Формується SQL-запит для вставки запису в таблицю `product_compare`. Використовується конструкція `INSERT INTO ... ON DUPLICATE KEY UPDATE`, щоб оновити запис, якщо він вже існує з такими ж ключами. Параметри `userId` і `productId` вставляються у відповідні місця запиту за допомогою підготовлених виразів, щоб запобігти SQL-ін'єкціям.

Запит виконується за допомогою методу `execute()` об'єкта `PDOStatement`. Результат виконання запиту повертається: `true`, якщо операція успішно виконана, або `false`, якщо сталася помилка.

3.3. Опис інструкції користувача

На першому етапі розробки онлайн-сервісу було важливо визначити його зовнішній вигляд. Одним з основних завдань було знайти шаблон зі зручним та простим інтерфейсом для користувача. Першочерговою вимогою до шаблону було наявність достатньої кількості блоків, до яких можна було б додати необхідну інформацію.

Для стилізації більшості елементів магазину було використано Bootstrap - відкритий і безкоштовний HTML, CSS та JS фреймворк. Bootstrap широко використовується веб-розробниками для швидкої верстки адаптивного дизайну сайтів і веб-додатків. Основним застосуванням цього фреймворку є розробка фронтенду сайтів і адміністративних панелей. Bootstrap є найпопулярнішим серед аналогів, оскільки дозволяє створювати верстку у декілька разів швидше. Він також легко доступний і вивчення [23].

У процесі розробки онлайн-сервісу виявлено, що елементи "header" та "footer" повторюються на більшості сторінок веб-ресурсу. Для оптимізації та скорочення коду було створено папку "layouts", в яку були додані файли "header.php" та "footer.php". Вони містять шляхи до CSS- та JS-файлів проекту та інші базові елементи веб-сайту.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		52

У представленнях, які використовують дані файли було додано наступний код:

```
<?php include ROOT . '/views/layouts/header.php'; ?>
```

```
<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Головна сторінка сайту (рис. 3.11) є ключовою для всіх інших розділів. Вона містить важливі посилання та інформацію, яка дозволяє користувачам виконувати необхідні дії, такі як авторизація, перегляд товарів або реєстрація.



Рисунок 3.11 – Головна сторінка сайту

Сторінка складається з декількох частин. Перша частина містить верхнє меню, авторизуватися в особистому кабінеті, переглянути кошик замовлень і оформити замовлення після авторизації, а також переглянути інформацію про компанію, та знайти інформацію як зв'язатись або знайти нас. Також у цій частині міститься основне меню, де є такі категорії, як букети, квіти поштучно, композиції, рослини в горщиках та подарунки. У цій частині розташований логотип сайту, який привертає увагу відвідувачів.

Друга частина містить рекомендовану продукцію сайту та нові товари.

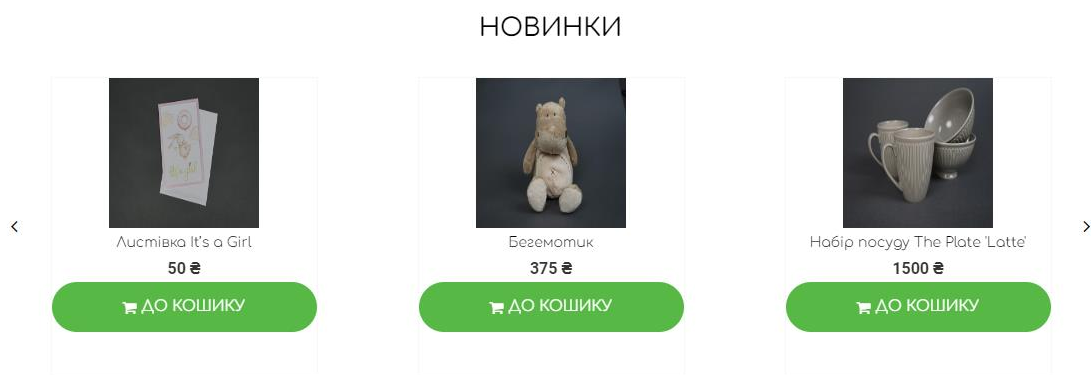


Рисунок 3.12 – Нові товари на головній сторінці

Третя частина – складається з лівої панелі з фільтрами, де можна здійснювати пошук за назвою, обрати категорію, розмір, сортувати за ціною та

рейтингом, а також встановити діапазон цін, та правої – де представлені усі товари на сайті.

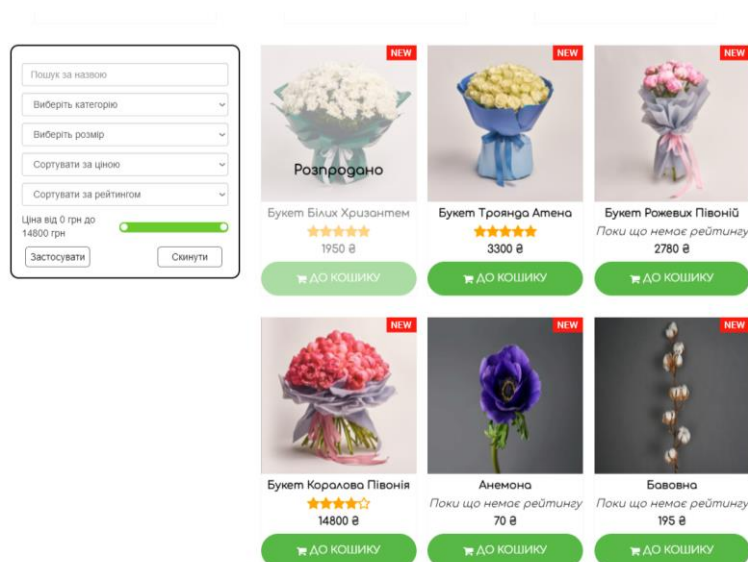


Рисунок 3.13 – Основна частина на головній сторінці

П'ята частина – це футер, який містить посилання на соціальні мережі, категорії сайту, перехід на вкладку «Про нас» та на «Контакти».



Рисунок 3.14 – Футер сайту

Натиснувши на кнопку «Акаунт» і обравши «Увійти» у верхньому меню головної сторінки, користувач бачить форму (рис. 3.15) для авторизації. При авторизації, після введення даних, запит надсилається на сервер, де здійснюється перевірка на рівні бази даних, і відповідно повертається відповідь. При успішній авторизації користувачу повертається сторінка з особистим кабінетом. Якщо дані введені некоректно, відображається повідомлення про помилку.

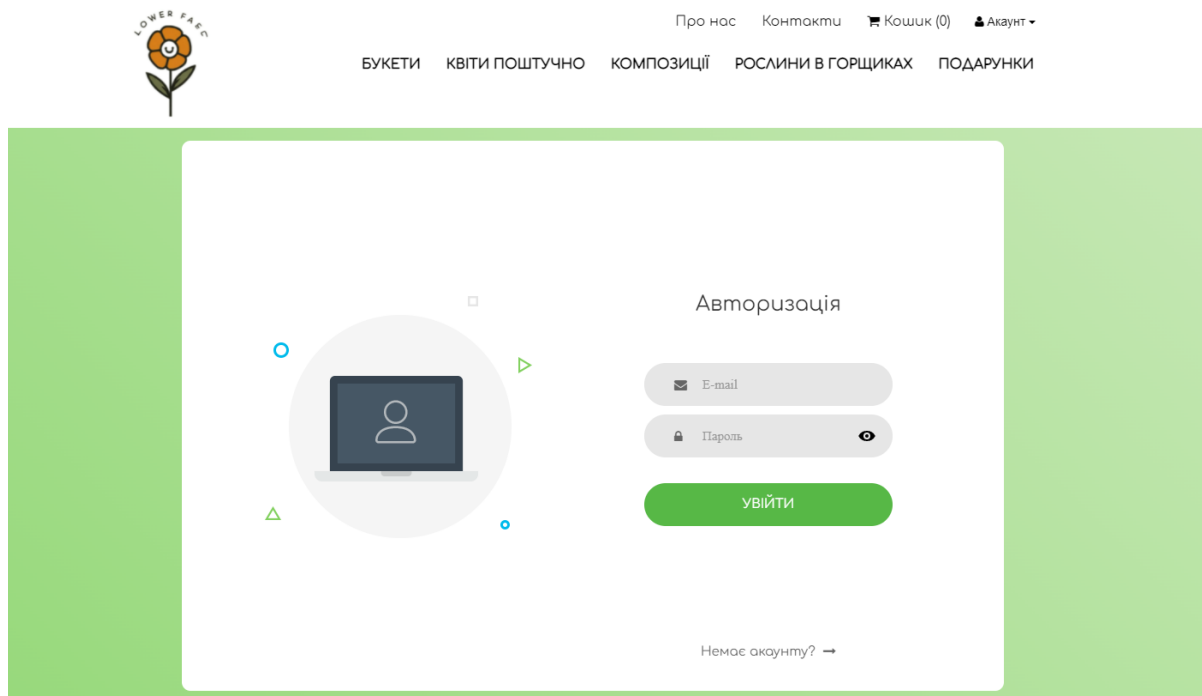


Рисунок 3.15 – Авторизація на сайті

Авторизовані користувачі на наступному етапі переходять до свого облікового запису (рис. 3.16).

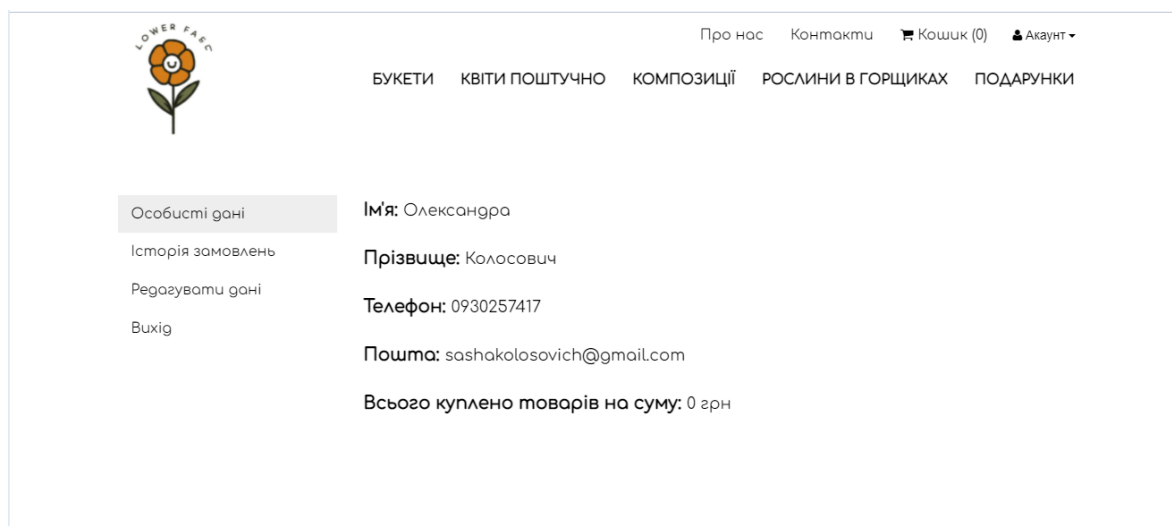


Рисунок 3.16 – Авторизація на сайті

За допомогою свого облікового запису клієнт може змінити основні дані про себе, пароль, переглянути історію замовлень



Про нас Контакти Кошик (0) Акаунт ▾

БУКЕТИ КВІТИ ПОШТУЧНО КОМПОЗИЦІЇ РОСЛИНИ В ГОРЩИКАХ ПОДАРУНКИ

Особисті дані
Історія замовлень
Редагувати дані
Вихід

Ім'я
Олександра

Прізвище
Колосович

Телефон
0930257417

Пошта
sashakolosovich@gmail.com

Пароль

Зберегти

Рисунок 3.17 – Зміна особистих даних



Про нас Контакти Кошик (0) Акаунт ▾

БУКЕТИ КВІТИ ПОШТУЧНО КОМПОЗИЦІЇ РОСЛИНИ В ГОРЩИКАХ ПОДАРУНКИ

Особисті дані
Історія замовлень
Редагувати дані
Вихід

ID замовлення	Телефон	Адреса	Дата оформлення	Статус	
31	0930257417		2022-07-19 18:26:06	Доставляється	

Рисунок 3.18 – Перегляд історії замовлень

На сайті передбачені закладки на улюблений товар (рис. 3.19), а також порівняння товарів.



Про нас Контакти Кошик (0) Акаунт ▾

БУКЕТИ КВІТИ ПОШТУЧНО КОМПОЗИЦІЇ РОСЛИНИ В ГОРЩИКАХ ПОДАРУНКИ

Ви додали до списку бажань

Видалити	Фото	Назва	Розмір	Ціна	Наявність
		Бавовна	S	195 ₴	Так
Очистити все					

Рисунок 3.19 – Список побажань

На головній сторінці розташовані всі товари магазину, поділеної на категорії. На цій сторінці доступні основні методи роботи з товарами, такі як

фільтр за ціною, фільтр за розміром, а також сортування за різними критеріями, наприклад, за ціною та датою оновлення товару (рис. 3.20).

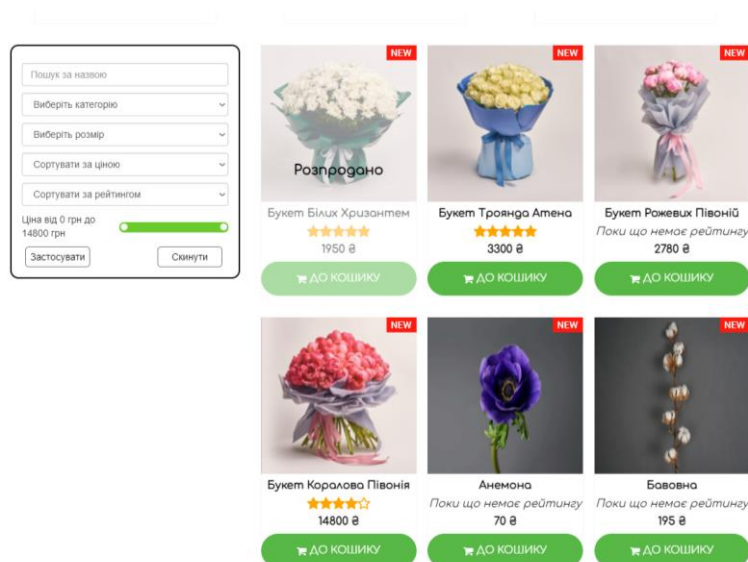


Рисунок 3.20 – Відображення товарів

Щоб отримати більш детальну інформацію про вибраний товар, потрібно натиснути на його назву. З даної сторінки користувачеві надається можливість(рис. 3.21):

- виставити рейтинг товару;
- написати відгук;
- додати до списку побажань;
- додати то списку порівняльної характеристики;
- додати до кошику.

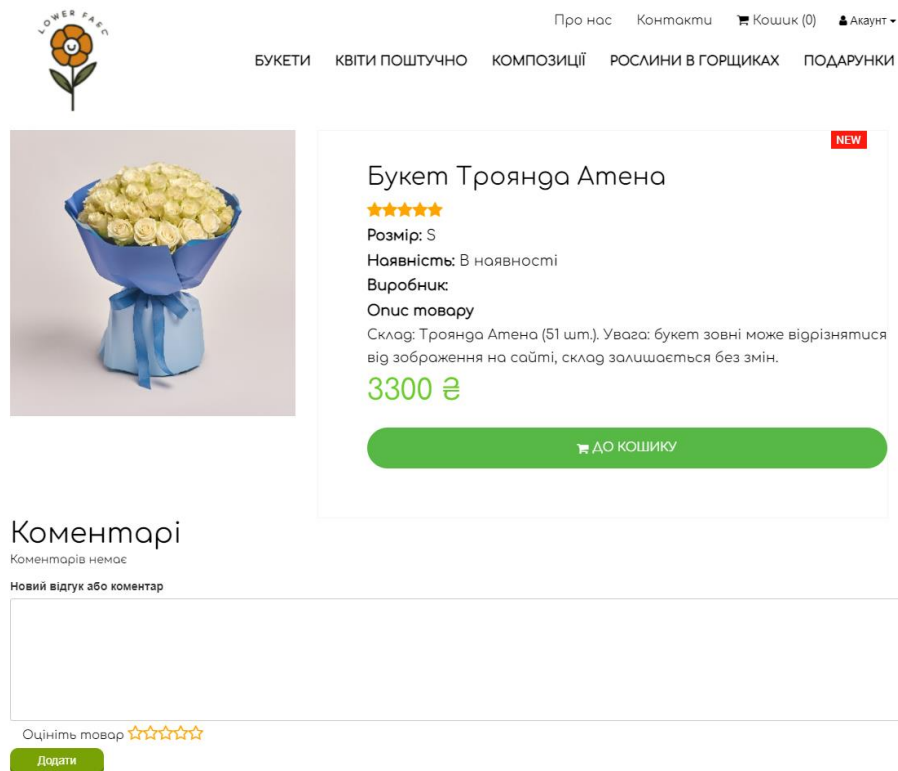


Рисунок 3.21 – Сторінка відображення детальної інформації про товар

Щоб оформити замовлення користувачу потрібно додати товар до кошику.

З даної сторінки користувачеві надається можливість(рис. 3.22):

- вибрати потрібну кількість;
- видалити товар з кошику;
- перейти до оформлення замовлення.

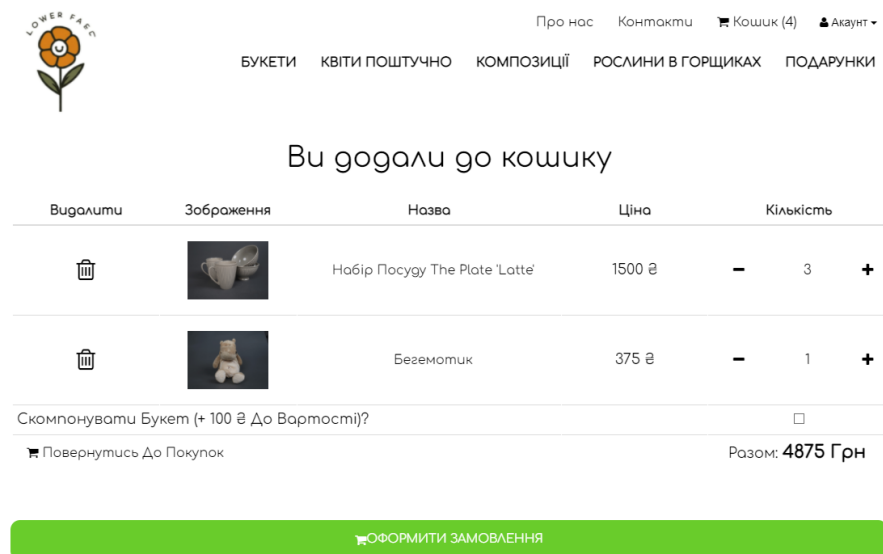


Рисунок 3.22 – Сторінка «Кошик»

Після додавання товарів до кошика, користувач натискає на кнопку «Оформити замовлення». Відкривається сторінка оформлення замовлення, де потрібно вказати контактні дані, адресу доставки та вибрати спосіб оплати, якщо користувач авторизований контактні дані, такі як ім'я, телефон будуть підтягнуті з бази даних. Після заповнення всіх полів користувач натискає кнопку «Підтвердити замовлення», і запит надсилається на сервер для обробки.

Рисунок 3.23 – Сторінка «Оформлення замовлення»

Окрім користувачів, не менш важливу роль відіграє адміністратор. Адміністратор контролює процес онлайн-торгівлі та працює над рейтингом інтернет-магазину, займається обліком відвідуваності, веде статистику кількості замовлень, коментарів тощо. Основним завданням адміністратора є наповнення контентом. Після авторизації на сайті як адміністратор з'являється додаткова вкладка «Адміністративна панель». Перейшовши до неї, адміністратор отримує доступ до ряду функцій (рис. 3.24).



← Повернутись на головну

Управління товарами
Управління категоріями
Управління бекапом
Статистика
Управління замовленнями
Управління зворотнім зв'язком

Рисунок 3.24 – Головна сторінка «Адмін панелі»

Для зручної роботи адміністратора з сайтом, було розроблено адміністративну панель за допомогою якої можна додати нові категорії (рис. 3.25), а відповідно до них товар (рис. 3.26). Окрім даної функції звісно ще можна переглядати, редагувати та видаляти.



← Повернутись на головну

Управління категоріями Додати категорію

Додати нову категорію

Назва

Порядковий номер

Зображення товару

Файл не выбран

Статус

Відображається ▾

Зберегти

Рисунок 3.25 – Сторінка додавання нової категорії

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		60



← Повернутись на головну

Новий товар

← Назад

Додати новий товар

Назва товару

Артикул

Вартість

Кількість

Розмір

Категорія

Рисунок 3.26 – Сторінка додавання товару

Виробник

Зображення товару

Детальний опис

Наявність на складі

Новинка

Рекомендовані

Статус

Зберегти

Рисунок 3.27 – Продовження сторінки додавання товару

Як зазвичай, в інтернет-магазині головним є клієнт та його замовлення, які обслуговує адміністратор. Основні функції, пов'язані із замовленнями, включають перегляд, редагування, видалення та зміну статусу замовлення (рис. 3.28).



[← Повернутись на головну](#)

Управління замовленнями

[← Назад](#)

Список замовлень

ID замовлення	Ім'я покупця	Телефон покупця	Адреса	Скомпонувати букет?	Тип оплати	Номер карти покупця	Дата оформлення	Статус			
44	Maria	0989574365	Короленка 87	Ні	Готівка		2024-05-28 12:46:39	Нове замовлення			
43	Sasha	0986525518	короленка 35	Ні	Готівка		2024-05-27 16:14:18	Нове замовлення			
42	Sasha	0986525518	короленка 35	Ні	Готівка		2024-05-27 16:05:12	Нове замовлення			
41	Sasha	0986525518	короленка 35	Ні	Готівка		2024-05-27 15:33:21	Нове замовлення			

Рисунок 3.28 – Сторінка замовлення

Реалізований інтерфейс для перегляду інформації з форми «Зворотній зв'язок» (рис. 3.29). З якої адміністратор аналізує роботу інтернет-магазину та виконує побажання клієнтів для підвищення рейтингу магазину.



[← Повернутись на головну](#)

Управління зворотнім зв'язком

[← Назад](#)

Зворотній зв'язок

ID	Ім'я	Пошта	Телефон	Текст	Дата	
14	Колосович Олександра	sashakolosovich@gmail.com	0930257417	Скільки днів триває гостівка	2022-07-19 18:18:28	

Рисунок 3.29 – Сторінка відображення запитань клієнтів

Також присутня роль Продавця–Флориста (рис. 3.30) який може керувати замовленнями та зворотнім зв'язком, та кур'єра (рис. 3.31), який може керувати тільки замовленнями.



[← Повернутись на головну](#)

Управління замовленнями

Управління зворотнім зв'язком

Рисунок 3.30 – Головна сторінка Продавця–Флориста

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		62



← Повернутись на головну

Управління замовленнями

← Назад

Список замовлень, які доставляються

ID замовлення	Ім'я покупця	Телефон покупця	Адреса	Скомпонувати букет?	Тип оплати	Номер карти покупця	Дата оформлення	Статус	
39	Софія	0985263518		Ні	Картка	4111111111111111	2022-07-22 17:50:25	Доставляється	👁
31	Олександра	0930257417		Ні	Картка	4111111111111111	2022-07-19 18:26:06	Доставляється	👁

Рисунок 3.31 – Головна сторінка Кур'єра

3.4. Тестування роботи програмного продукту

Створення будь-якого програмного продукту неминуче супроводжується деякими помилками в процесі розробки, тому тестування програмного забезпечення є необхідним для забезпечення високої якості додатка. Під час тестування перевіряються різні аспекти програми, такі як коректність вхідних та вихідних даних, правильність роботи функцій та алгоритмів, обробка помилок і відновлення після збоїв. Метою тестування є виявлення помилок і дефектів програмного продукту, щоб їх можна було виправити до випуску на ринок або введення в експлуатацію [21].

Спершу створимо так звані Test Cases [16]. Test Case є тестовим артефактом, який полягає у виконанні певної кількості дій та/або умов, необхідних для перевірки визначеної функціональності розроблюваної програмної системи. Представимо створені тестові випадки у вигляді таблиці.

Таблиця 3.1

Test Cases до онлайн сервісу доставки квітів

Test Case Description	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
1.	2.	3.	4.	5.	6.
Перевірка авторизації з дійсними даними	1. Перейти на сторінку http://flowers1/user/login/ 2. Введіть email 3. Введіть пароль 4. Натисніть "Увійти"	Email = admin@gmail.com Пароль = 1111111111	Користувач повинен увійти в систему	Як і очікувалося	Pass
Перевірка авторизації з недійсними даними	1. Перейти на сторінку http://flowers1/user/login/ 2. Введіть email 3. Введіть пароль 4. Натисніть "Увійти"	Email = admin@gmail.com Пароль = 12345678	Користувач не повинен увійти в систему і отримати відповідне повідомлення	Як і очікувалося	Pass
Перевірка успішного оформлення замовлення	1. Перейти на сторінку http://flowers1 2. Вибрати букет квітів 3. Додати букет до кошика 4. Перейти до кошика 5. Натиснути "Оформити замовлення"	Ім'я = Олексій Адреса = вул. Шевченка, 10 Телефон = 0987654321 Картка = 1234 5678 9012 3456	Замовлення повинно бути успішно оформлене, показано повідомлення про успіх	Як і очікувалося	Pass

Продовження таблиці 3.1

1.	2.	3.	4.	5.	6.
	6. Ввести дані для доставки (ім'я, адреса, телефон) 7. Вибрати спосіб оплати (картка) 8. Натиснути "Підтвердити замовлення"				
Перевірка оформлення замовлення без введення даних для доставки	1. Перейти на сторінку http://flowers1 2. Вибрати букет квітів 3. Додати букет до кошика 4. Перейти до кошика 5. Натиснути "Оформити замовлення" 6. Не вводити дані для доставки 7. Вибрати спосіб оплати (готівка) 8. Натиснути "Підтвердити замовлення"	—	Система повинна вивести повідомлення про необхідність введення даних для доставки	Як і очікувалося	Pass
Перевірка пошуку товару за категорією	1. Перейти на головну сторінку http://flowers1 2. Обрати категорію "Букети"	Категорія = Букети	Повинен відобразитися список товарів з	Як і очікувалося	Pass

Закінчення таблиці 3.1

1.	2.	3.	4.	5.	6.
	3. Натиснути "Застосувати"		категорії "Букети"		
Перевірка пошуку товару за категорією	1. Перейти на головну сторінку http://flowers1 2. Обрати категорію "Букети" 3. Натиснути "Застосувати"	Категорія = Букети	Повинен відобразитися список товарів з категорії "Букети"	Як і очікувалося	Pass

В ході проведення тестування було перевірено додаток та виправлено знайдені помилки. В результаті отримано застосунок відповідної якості.

Висновки до третього розділу

В даному розділі було детально описано архітектуру системи, що базується на шаблоні MVC (Модель-Вид-Контролер), що забезпечує чітке розділення логіки, даних і представлення. Особлива увага приділена функціональності для користувачів та адміністраторів. Користувачам надано можливості для авторизації, реєстрації, перегляду товарів, додавання їх до корзини та списку побажань. Адміністратори мають доступ до розширених функцій через адміністративну панель, що дозволяє контролювати процеси торгівлі, вести облік відвідувань, замовлень та коментарів, а також наповнювати сайт контентом. Важливість тестування підкреслюється через наведення прикладів тестових сценаріїв для перевірки основних функцій.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було досягнуто цілей, поставлених на початку дослідження. Було розроблено автоматизовану систему підбору букетів для квіткового магазину, яка суттєво спрощує процес вибору товару для клієнтів і підвищує ефективність роботи магазину.

На першому етапі роботи проведено детальний аналіз технічного завдання і визначено вимоги до програмного продукту. Це включало вибір необхідних засобів моделювання, створення логічної моделі системи та діаграм, що ілюструють алгоритми роботи та функціональні взаємозв'язки всіх елементів системи.

Далі було реалізовано інтерфейс користувача та адміністративну панель, що забезпечує зручний доступ до всіх функцій системи як для клієнтів, так і для адміністратора. Адміністративна панель дозволяє ефективно керувати замовленнями, товаром і користувачами, а також вести облік і статистику діяльності магазину

Окрему увагу приділено тестуванню розробленого програмного забезпечення. Створено та проведено різні тестові випадки для перевірки коректності роботи системи. Це дозволило виявити та усунути помилки, забезпечивши високу якість і надійність кінцевого продукту.

Впровадження даної автоматизованої системи дозволяє зменшити час на обслуговування клієнтів, підвищити продуктивність працівників магазину та мінімізувати людські помилки. Крім того, це сприяє покращенню загального рівня обслуговування клієнтів, що позитивно впливає на рейтинг і конкурентоспроможність квіткового магазину на ринку.

Отже, результатом виконання кваліфікаційної роботи є розроблений програмний продукт, який відповідає всім визначеним вимогам, практичному значені, що підвищує конкурентоспроможність компанії, і готовий до подальшої реалізації та використання в реальних умовах.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		67

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Васильєв О.М. Програмування мовою PHP / Ліра-К, 2022. – 368 с.
2. Веб-технології та веб-дизайн / О. Г. Трофименко, О. Б. Козін, О. В. Задерейко, О. Є. Плачинда., 2019. – 284 с.
3. Використання технологій jQuery, Ajax при розробці інтернет-сайтів. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.kursak.com/vykorystannya-tehnolohij-jquery-ajax-pry-rozrobtisi-internet-sajtiv/>.
4. Діаграма діяльності. [Електронний ресурс] – Режим доступу до ресурсу: <https://studfile.net/preview/5200239/page:7/>.
5. Доставка і замовлення квітів по всій Україні - Флора24. [Електронний ресурс] – Режим доступу до ресурсу: <http://flora24.com.ua>
6. Доставка квітів – актуальність і розвиток незвичайної послуги. [Електронний ресурс] – Режим доступу до ресурсу: <https://bukinfo.com.ua/vidpochynok/dostavka-kvitiv-aktualnist-i-rozvytok-nezvychaynoji-poslugy>.
7. Квіти з доставкою – зручний сервіс чи необхідність? [Електронний ресурс] – Режим доступу до ресурсу: <https://landlord.ua/news/novyny-partneriv/kvityi-z-dostavkoyu-zruchnyj-servis-chy-neobhidnist/>.
8. Методи передачі даних (GET і POST). [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.php.brj.cz/metodi-peredaci-danih-get-i-post>.
9. Офіційна документація MySQL. [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/>
10. Офіційна документація PHP. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.php.net/manual/en/function.phpinfo.php>.
11. Підручник HTML. [Електронний ресурс] – Режим доступу до ресурсу: https://htmlbook.at.ua/news/tutorial_html/1-0-1.

12. Побудова динамічних вебзастосунків за допомогою MVC. [Електронний ресурс] – Режим доступу до ресурсу: <https://blog.ithillel.ua/articles/building-dynamic-web-applications-using-mvc>
13. Служба доставки квітів Flowers De Lux. [Електронний ресурс] – Режим доступу до ресурсу: <https://flowersdelux.com.ua>
14. Сучасний підручник з JavaScript. [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.javascript.info>.
15. Сучасний підручник та довідник з CSS. [Електронний ресурс] – Режим доступу до ресурсу: <https://css.tutorial.in.ua>.
16. Тест-кейс – один із ключових інструментів тестувальників ПЗ. [Електронний ресурс] – Режим доступу до ресурсу: <https://foxminded.ua/test-keis/>.
17. Тест-кейси: що, як, навіщо. [Електронний ресурс] – Режим доступу до ресурсу: <https://dou.ua/forums/topic/46769/>.
18. Шаховська Н.Б., Литвин В.В. Проектування інформаційних систем: навч. посіб. Львів: «Магнолія – 2006», 2011. – 380с.
19. Юрій Грицюк. Аналіз вимог до програмного забезпечення / Юрій Грицюк., 2018. – 456 с.
20. Як відправляти листи за допомогою PHPMailer. [Електронний ресурс] – Режим доступу до ресурсу: <https://sendpulse.ua/knowledge-base/smtp/smtp-sendpulse-phpmailer>.
21. Як тестувати веб-сайт: основні етапи і поради. [Електронний ресурс] – Режим доступу до ресурсу: <https://brainlab.com.ua/uk/blog-uk/yak-testuvati-veb-sayt-osnovn-etapi-poradi>.
22. Bloom квіткова майстерня. [Електронний ресурс] – Режим доступу до ресурсу: <https://bloom.ua>.
23. Bootstrap 4 Підручник. [Електронний ресурс] – Режим доступу до ресурсу: <https://w3schoolsua.github.io/bootstrap4/index.html#gsc.tab=0>.
24. Introduction. StarUML documentation. [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.staruml.io>.

25. Jon Duckett. HTML and CSS: Design and Build Websites / Jon Duckett., 2011. – (512).
26. Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition / Martin Fowler., 2016. – 208 с.
27. PHP Підручник. [Електронний ресурс] – Режим доступу до ресурсу: <https://w3schoolsua.github.io/php/index.html#gsc.tab=0>.
28. SciencSoft. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.scnsoft.com/blog/web-application-architecture>.
29. UML для бізнес-моделювання: для чого потрібні діаграми процесів. [Електронний ресурс] – Режим доступу до ресурсу: <https://evergreens.com.ua/ua/articles/uml-diagrams.html>.
30. What is Class Diagram. Visual Paradigm. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/whatis-class-diagram/>.

ДОДАТКИ

Варіант використання: Реєстрація користувача

Ім'я варіантів використання: Реєстрація користувача		Рівень важливості: високий
Основний актор: Користувач		
Інші учасники: Відсутні		
Короткий опис:	Користувач реєструє новий обліковий запис на веб-сайті, надаючи необхідну інформацію.	
Передумови:	Користувач повинен мати доступ до інтернету і не бути зареєстрованим на сайті.	
Основний потік:	1. Користувач натискає на кнопку "Реєстрація". 2. Система відображає форму реєстрації. 3. Користувач заповнює поля форми: ім'я, прізвище, номер телефону, електронна пошта, пароль. 4. Користувач натискає кнопку "Зареєструватися". 5. Система перевіряє правильність введених даних та унікальність електронної пошти. 6. Система зберігає дані нового користувача у базі даних. 7. Система відправляє користувачу підтвердження реєстрації на вказану електронну пошту. 8. Система інформує користувача про успішну реєстрацію та перенаправляє на сторінку входу.	
Альтернативні / виключні потоки:	A1: Користувач вводить некоректні дані (наприклад, паролі не співпадають або електронна пошта вже існує). 1. Система відображає повідомлення про помилку та пропонує виправити введені дані.	

Варіант використання: Зміна персональних даних

Ім'я варіантів використання: Зміна персональних даних		Рівень важливості: середній
Основний актор: Користувач		
Інші учасники: Відсутні		
Короткий опис:	Користувач змінює свої персональні дані, такі як ім'я, адреса, телефон тощо, у своєму обліковому записі.	
Передумови:	Користувач повинен бути зареєстрований та увійти в систему.	
Основний потік:	1. Користувач входить у свій обліковий запис. 2. Користувач переходить до розділу "Персональні дані". 3. Система відображає поточні дані користувача. 4. Користувач вносить зміни до необхідних полів (ім'я, номер телефону, адреса тощо). 5. Користувач натискає кнопку "Зберегти зміни". 6. Система перевіряє правильність введених даних. 7. Система зберігає оновлені дані у базі даних. 8. Система відображає повідомлення про успішне збереження змін.	
Альтернативні / виключні потоки:	A1: Користувач вводить некоректні дані (наприклад, неправильний формат електронної пошти). 1. Система відображає повідомлення про помилку .	

Варіант використання: Залишити відгук

Ім'я варіантів використання: Залишити відгук		Рівень важливості: середній
Основний актор: Користувач		
Інші учасники: Відсутні		
Короткий опис:	Авторизований користувач залишає відгук про товар або послугу на веб-сайті.	
Передумови:	Користувач повинен бути зареєстрований та увійти в систему.	
Основний потік:	1. Користувач входить у свій обліковий запис. 2. Користувач переходить на сторінку товару, про яку хоче залишити відгук. 3. Користувач пише відгук у відповідне поле. 4. Користувач натискає на кнопку "Залишити відгук". 5. Система зберігає відгук у базі даних.	
Альтернативні / виключні потоки:	A1: Користувач не авторизований. 1. Система відображає повідомлення про необхідність авторизації для залишення відгуку.	

Варіант використання: Додавання категорії

Ім'я варіантів використання: Додавання категорії		Рівень важливості: високий
Основний актор: Адміністратор		
Інші учасники: Відсутні		
Короткий опис:	Адміністратор додає нову категорію до каталогу на веб-сайті, вказуючи її назву та опис.	
Передумови:	Адміністратор повинен бути авторизований у системі.	
Основний потік:	1. Адміністратор входить у свій обліковий запис. 2. Адміністратор переходить до розділу управління категоріями. 3. Адміністратор натискає на кнопку "Додати нову категорію". 4. Система відображає форму для додавання нової категорії. 5. Адміністратор заповнює поля форми: назва категорії, опис. 6. Адміністратор натискає кнопку "Зберегти". 7. Система перевіряє правильність введених даних. 8. Система зберігає нову категорію у базі даних. 9. Система відображає повідомлення про успішне додавання категорії та оновлює каталог.	
Альтернативні / виключні потоки:	A1: Адміністратор вводить некоректні дані (наприклад, відсутність обов'язкових полів). 1. Система відображає повідомлення про помилку та пропонує виправити введені дані.	

Варіант використання: Додавання букету

Ім'я варіантів використання: Додавання букету		Рівень важливості: високий
Основний актор: Адміністратор		
Інші учасники: Відсутні		
Короткий опис:	Адміністратор додає новий букет до каталогу на веб-сайті, вказуючи його назву, опис, ціну та інші деталі.	
Передумови:	Адміністратор повинен бути авторизований у системі.	
Основний потік:	1. Адміністратор входить у свій обліковий запис. 2. Адміністратор переходить до розділу управління товарами. 3. Адміністратор натискає на кнопку "Додати новий букет". 4. Система відображає форму для додавання нового букету. 5. Адміністратор заповнює поля форми: назва букету, опис, ціна, зображення, категорія тощо. 6. Адміністратор натискає кнопку "Зберегти". 7. Система перевіряє правильність введених даних. 8. Система зберігає новий букет у базі даних. 9. Система відображає повідомлення про успішне додавання букету та оновлює каталог.	
Альтернативні / виключні потоки:	A1: Адміністратор вводить некоректні дані (наприклад, відсутність обов'язкових полів). 1. Система відображає повідомлення про помилку та пропонує виправити введені дані.	

Варіант використання: Управління замовленнями

Ім'я варіантів використання: Управління замовленнями		Рівень важливості: високий
Основний актор: Продавець-Флорист, Адміністратор, Кур'єр		
Інші учасники: Відсутні		
Короткий опис:	Продавець-Флорист, Адміністратор або Кур'єр переглядає, оновлює та керує замовленнями користувачів на веб-сайті.	
Передумови:	Продавець-Флорист, Адміністратор або Кур'єр повинен бути авторизований у системі.	
Основний потік:	1. Продавець-Флорист, Адміністратор або Кур'єр входить у свій обліковий запис. 2. Продавець-Флорист, Адміністратор або Кур'єр переходить до розділу управління замовленнями. 3. Система відображає список усіх замовлень 4. Продавець-Флорист, Адміністратор або Кур'єр вибирає конкретне замовлення для перегляду деталей. 5. Система відображає деталі обраного замовлення. 6. Продавець-Флорист, Адміністратор або Кур'єр може оновити статус замовлення (наприклад, "в обробці", "відправлено", "доставлено") або внести інші зміни. 7. Продавець-Флорист, Адміністратор або Кур'єр натискає кнопку "Зберегти зміни". 8. Система зберігає оновлені дані у базі даних. 9. Система відображає повідомлення про успішне збереження змін та оновлює список замовлень.	
Альтернативні / виключні потоки:	A1: Продавець-Флорист, Адміністратор або Кур'єр вводить некоректні дані. 1. Система відображає повідомлення про помилку та пропонує виправити введені дані.	

Контролер «CatalogController»

```

<?php
class CatalogController
{
    public function actionIndex()
    {
        $categories = Category::getCategoriesList();
        $latestProducts = Product::getLatestProducts(50);
        require_once(ROOT . '/views/catalog/index.php');
        return true;
    }
    public function actionCategory($categoryId, $page = 1)
    {
        $categories = array();
        $categories = Category::getCategoriesList();
        foreach ($categories as $categoryItem) {
            if ($categoryItem['id'] == $categoryId) {
                $category = $categoryItem;
                break;
            }
        }
        $search = $_GET['search'] ?? '';
        $price_sort = $_GET['price_sort'] ?? '';
        $rating_sort = $_GET['rating_sort'] ?? '';
        $price_min = $_GET['price_min'] ?? '';
        $price_max = $_GET['price_max'] ?? '';
        $size = $_GET['size'] ?? '';
        $params = array(
            'categoryId' => $categoryId,
            'search' => $search,
            'page' => $page,
            'price_sort' => $price_sort,
            'rating_sort' => $rating_sort,
            'price_min' => $price_min,
            'price_max' => $price_max,
            'size' => $size,
            'hasPagination' => true
        );
        $categoryProducts = Product::getProducts($params);
        $userId = User::userId();
        if (isset($userId)) {
            $wishlist = Wishlist::getWishlist($userId);
            $productCompare = ProductCompare::getProductCompare($userId);
        }
        $maxPrice = Product::getMaxPrice($categoryId);
        $total = Product::getTotalProductsInCategory($categoryId, $search);
        $pagination = new Pagination($total, $page, Product::SHOW_BY_DEFAULT, 'page-');
        require_once(ROOT . '/views/catalog/category.php');
        return true;
    }
    public function actionDodat()
    {
        $categories = array();
        $categories = Category::getCategoriesList();
        require_once(ROOT . '/views/catalog/dodat.php');
        return true;
    }
}

```

Модель «Product»

```

<?php
class Product
{
    const SHOW_BY_DEFAULT = 6;
    public static function checkProductsQuantity($products)
    {
        $db = Db::getConnection();
        $query = 'SELECT * FROM product';
        $productsConditions = array();
        $i = 0;
        foreach ($products as $productId => $quantity){
            $productsConditions[$i++] = '(id = \'' . $productId . '\' AND quantity < \'' . $quantity
            . '\')';
        }
        if (count($productsConditions) == 0) {
            return false;
        }
        $query .= ' WHERE ' . implode(' OR ', $productsConditions);
        $query .= ' LIMIT 1';
        $result = $db->query($query);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        while ($result->fetch()) {
            return false;
        }
        return true;
    }
    public static function subtractProductsQuantity($products)
    {
        $db = Db::getConnection();
        foreach ($products as $productId => $quantity) {
            $query = 'UPDATE product SET quantity = (case when (quantity - :quantity) < 0 then 0
            else (quantity - :quantity) end) WHERE id = :productId';
            $result = $db->prepare($query);
            $result->bindParam(':productId', $productId, PDO::PARAM_INT);
            $result->bindParam(':quantity', $quantity, PDO::PARAM_INT);
            $result->execute();
        }
    }
    public static function getLatestProducts($count = self::SHOW_BY_DEFAULT)
    {
        $db = Db::getConnection();
        $sql = 'SELECT *, ' . self::getRatingAttr() . ' FROM product '
            . 'WHERE status = "1" ORDER BY id DESC '
            . 'LIMIT :count';
        $result = $db->prepare($sql);
        $result->bindParam(':count', $count, PDO::PARAM_INT);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $result->execute();
        $i = 0;
        $productsList = array();
        while ($row = $result->fetch()) {
            $productsList[$i++] = $row;
        }
        return $productsList;
    }
    public static function getRatingAttr()
    {
        return '(select round(avg(rate), 2) from product_rating where product_rating.product_id =
        product.id) as rating';
    }
    public static function getProducts($params)

```

```

{
    $categoryId = $params['categoryId'];
    $search = $params['search'];
    $page = $params['page'] ?? 1;
    $price_sort = $params['price_sort'];
    $rating_sort = $params['rating_sort'] ?? '';
    $price_min = $params['price_min'];
    $price_max = $params['price_max'];
    $size = $params['size'];
    $hasPagination = $params['hasPagination'] ?? true;
    $db = Db::getConnection();
    $sql = 'SELECT *, ' . self::getRatingAttr() . ' FROM product WHERE status = 1';
    $order = array();
    if ($search != '') {
        $sql .= " AND trim(lower(name)) LIKE lower('%" . trim($search) . "%')";
    }

    if ($categoryId != '') {
        $sql .= " AND category_id = :category_id";
    }
    if ($size != '') {
        $sql .= " AND size = :size";
    }
    if ($price_min != '') {
        $sql .= " AND price >= :price_min";
    }
    if ($price_max != '') {
        $sql .= " AND price <= :price_max";
    }
    if ($rating_sort != '') {
        $order[] = "rating $rating_sort";
    }
    if ($price_sort != '') {
        $order[] = "price $price_sort";
    } else {
        $order[] = "id ASC";
    }
    $sql .= " ORDER BY " . implode(", ", $order);
    if ($hasPagination) {
        $sql .= " LIMIT :limit OFFSET :offset";
    }
    $result = $db->prepare($sql);
    if ($categoryId != '') {
        $result->bindParam(':category_id', $categoryId, PDO::PARAM_INT);
    }
    if ($size != '') {
        $result->bindParam(':size', $size, PDO::PARAM_STR);
    }
    if ($price_min != '') {
        $result->bindParam(':price_min', $price_min, PDO::PARAM_INT);
    }
    if ($price_max != '') {
        $result->bindParam(':price_max', $price_max, PDO::PARAM_INT);
    }
    if ($hasPagination) {
        $limit = Product::SHOW_BY_DEFAULT;
        $offset = ($page - 1) * self::SHOW_BY_DEFAULT;
        $result->bindParam(':limit', $limit, PDO::PARAM_INT);
        $result->bindParam(':offset', $offset, PDO::PARAM_INT);
    }
    $result->execute();
    $i = 0;
    $products = array();
    while ($row = $result->fetch()) {
        $products[$i++] = $row;
    }
}

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		80

```

        return $products;
    }
    public static function getMaxPrice($categoryId = null)
    {
        $db = Db::getConnection();
        $sql = 'SELECT max(price) as max_price FROM product WHERE status = 1';
        if (isset($categoryId)) {
            $sql .= " AND category_id = " . "'" . $categoryId . "'";
        }
        $result = $db->prepare($sql);
        $result->execute();
        while ($row = $result->fetch()) {
            return $row['max_price'] ?? 0;
        }
    }
    public static function getProductById($id, $includeComments = false)
    {
        $db = Db::getConnection();
        if ($includeComments) {
            $sql = '
SELECT product.*,
        ' . self::getRatingAttr() . ',
        product_comment.id as comment_id,
        product_comment.comment,
        product_comment.created_at,
        user.id as user_id,
        user.name as user_name,
        user.lastname,
        user.role
FROM product
        LEFT JOIN product_comment on product.id = product_comment.product_id
        LEFT JOIN user on product_comment.user_id = user.id';
        } else {
            $sql = 'SELECT *, ' . self::getRatingAttr() . ' FROM product';
        }
        $sql .= ' WHERE product.id = :id';
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->execute();
        $i = 0;
        $product = null;
        while ($row = $result->fetch()) {
            //if (!$product) $product = array(...$row);
            if (!$product) $product = $row;
            if (!$includeComments) break;

            $product['comments'] = $product['comments'] ?? array();
            if (!isset($row['comment_id'])) break;
            $product['comments'][$i++] = array(
                'id' => $row['comment_id'],
                'comment' => $row['comment'],
                'created_at' => $row['created_at'],
                'user' => array(
                    'id' => $row['user_id'],
                    'name' => $row['user_name'],
                    'lastname' => $row['lastname'],
                    'role' => $row['role']
                )
            );
        }
        return $product;
    }
    public static function getTotalProductsInCategory($categoryId, $search = '')
    {
        $db = Db::getConnection();
        $sql = 'SELECT count(id) AS count FROM product WHERE status="1" AND category_id = :category_id';

```

```

        if ($search != '') {
            $sql .= " AND trim(lower(name)) LIKE lower('%" . trim($search) . "%')";
        }
        $result = $db->prepare($sql);
        $result->bindParam(':category_id', $categoryId, PDO::PARAM_INT);
        $result->execute();
        $row = $result->fetch();
        return $row['count'];
    }
    public static function getProductsByIds($idsArray)
    {
        $db = Db::getConnection();
        $idsString = implode(',', $idsArray);
        $sql = "SELECT *, " . self::getRatingAttr() . " FROM product WHERE status='1' AND id IN
($idsString)";
        $result = $db->query($sql);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $i = 0;
        $products = array();
        while ($row = $result->fetch()) {
            $products[$i++] = $row;
        }
        return $products;
    }
    public static function getRecommendedProducts()
    {
        $db = Db::getConnection();
        $result = $db->query('SELECT *, ' . self::getRatingAttr() . ' FROM product '
            . 'WHERE status = "1" AND is_recommended = "1" '
            . 'ORDER BY id DESC');
        $i = 0;
        $productsList = array();
        while ($row = $result->fetch()) {
            $productsList[$i++] = $row;
        }
        return $productsList;
    }
    public static function getProductsList()
    {
        $db = Db::getConnection();
        $result = $db->query('SELECT *, ' . self::getRatingAttr() . ' FROM product ORDER BY id
ASC');
        $productsList = array();
        $i = 0;
        while ($row = $result->fetch()) {
            $productsList[$i++] = $row;
        }
        return $productsList;
    }
    public static function deleteProductById($id)
    {
        $db = Db::getConnection();
        $sql = 'DELETE FROM product WHERE id = :id';
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        return $result->execute();
    }
    public static function updateProductById($id, $options)
    {
        $db = Db::getConnection();
        $sql = "UPDATE product
            SET
                name = :name,
                category_id = :category_id,
                code = :code,
                price = :price,

```

```

        availability = :availability,
        quantity = :quantity,
        brand = :brand,
        description = :description,
        is_new = :is_new,
        size = :size,
        is_recommended = :is_recommended,
        status = :status
    WHERE id = :id";
$result = $db->prepare($sql);
$result->bindParam(':id', $id, PDO::PARAM_INT);
$result->bindParam(':name', $options['name'], PDO::PARAM_STR);
$result->bindParam(':code', $options['code'], PDO::PARAM_STR);
$result->bindParam(':price', $options['price'], PDO::PARAM_STR);
$result->bindParam(':category_id', $options['category_id'], PDO::PARAM_INT);
$result->bindParam(':brand', $options['brand'], PDO::PARAM_STR);
$result->bindParam(':availability', $options['availability'], PDO::PARAM_INT);
$result->bindParam(':quantity', $options['quantity'], PDO::PARAM_INT);
$result->bindParam(':description', $options['description'], PDO::PARAM_STR);
$result->bindParam(':is_new', $options['is_new'], PDO::PARAM_INT);
$result->bindParam(':is_recommended', $options['is_recommended'], PDO::PARAM_INT);
$result->bindParam(':size', $options['size'], PDO::PARAM_STR);
$result->bindParam(':status', $options['status'], PDO::PARAM_INT);
return $result->execute();
}

public static function createProduct($options)
{
    $db = Db::getConnection();
    $sql = 'INSERT INTO product '
        . '(name, category_id, code, price, quantity, availability, brand,'
        . 'description, is_new, is_recommended, status, size)'
        . 'VALUES '
        . '(:name, :category_id, :code, :price, :quantity, :availability, :brand,'
        . ':description, :is_new, :is_recommended, :status, :size)';
    $result = $db->prepare($sql);
    $result->bindParam(':name', $options['name'], PDO::PARAM_STR);
    $result->bindParam(':category_id', $options['category_id'], PDO::PARAM_INT);
    $result->bindParam(':code', $options['code'], PDO::PARAM_STR);
    $result->bindParam(':price', $options['price'], PDO::PARAM_STR);
    $result->bindParam(':availability', $options['availability'], PDO::PARAM_INT);
    $result->bindParam(':brand', $options['brand'], PDO::PARAM_STR);
    $result->bindParam(':description', $options['description'], PDO::PARAM_STR);
    $result->bindParam(':is_new', $options['is_new'], PDO::PARAM_INT);
    $result->bindParam(':is_recommended', $options['is_recommended'], PDO::PARAM_INT);
    $result->bindParam(':size', $options['size'], PDO::PARAM_STR);
    $result->bindParam(':status', $options['status'], PDO::PARAM_INT);
    $result->bindParam(':quantity', $options['quantity'], PDO::PARAM_INT);
    if ($result->execute()) {
        return $db->lastInsertId();
    }
    return 0;
}

public static function getAvailabilityText($availability)
{
    switch ($availability) {
        case '1':
            return 'В наявності';
            break;
        case '0':
            return 'Під замовлення';
            break;
    }
}

public static function getImages($id)
{
    $path = '/upload/images/products/';

```

```

    $files = scandir($_SERVER['DOCUMENT_ROOT'] . $path);
    $image = Product::getImage($id);
    $images = array(
        $image => $image
    );
    foreach ($files as $file) {
        $path = '/upload/images/products/' . $file;
        if (preg_match("~^$id(\\.\\d+)?\\.jpg$~", $file) && $path != $image) {
            $images[$path] = $path;
        }
    }
    return $images;
}
public static function getImage($id)
{
    $noImage = 'no-image.jpg';
    $path = '/upload/images/products/';
    $pathToProductImage = $path . $id . '.jpg';
    if (file_exists($_SERVER['DOCUMENT_ROOT'] . $pathToProductImage)) {
        return $pathToProductImage;
    }
    return $path . $noImage;
}
}

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		84

Лістинг cabinet\index.php

```

<?php include ROOT . '/views/layouts/header.php'; ?>
<link href="/template/css/cabinet.css" rel="stylesheet">
<section class="main_content_area">
    <div class="container">
        <div class="account_dashboard">
            <div class="row">
                <div class="col-sm-10 col-md-3 ">
                    <!-- Nav tabs -->
                    <div class="dashboard_tab_button">
                        <ul role="tablist" class="nav flex-column dashboard-list">
                            <li class="active"><a href="#dashboard" data-toggle="tab"
class="nav-link">Особисті
                                дані</a></li>
                            <li><a href="#orders" data-toggle="tab" class="nav-link">Історія
замовлень</a></li>
                            <li><a href="#account-details" data-toggle="tab" class="nav-
link">Редагувати дані</a>
                                </li>
                            <li><a href="/user/login/" class="nav-link">Вихід</a></li>
                        </ul>
                    </div>
                </div>
                <div class="col-sm-10 col-sm-7">
                    <div class="tab-content dashboard_content">
                        <div class="tab-pane fade show active in" id="dashboard">
                            <p><span class="info_span"> Ім'я: </span> <?php echo
$user['name']; ?> </p>
                            <br>
                            <p><span class="info_span"> Прізвище: </span> <?php echo
$user['lastname']; ?> </p>
                            <br>
                            <p><span class="info_span"> Телефон: </span> <?php echo
$user['phone']; ?> </p>
                            <br>
                            <p><span class="info_span"> Пошта: </span> <?php echo
$user['email']; ?> </p>
                            <br>
                            <p><span class="info_span"> Всього куплено товарів на суму:
</span> <?php echo $totalPrice; ?> грн </p>
                        </div>
                        <div class="tab-pane fade" id="orders">
                            <div class="container">
                                <div class="row">
                                    <table class="table-bordered table-striped table">
                                        <tr>
                                            <th class="table-admin">ID замовлення</th>
                                            <th class="table-admin">Телефон</th>
                                            <th class="table-admin">Адреса</th>
                                            <th class="table-admin">Дата оформлення</th>
                                            <th class="table-admin">Статус</th>
                                        </tr>
                                        <tr>
                                            <td>
                                                <a href="/admin/order/view/<?php echo
$order['id']; ?>">
                                                    <?php echo $order['id']; ?>
                                                </a>
                                            </td>
                                            <td><?php echo $order['user_phone']; ?></td>
                                        </tr>
                                    </table>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>

```

```

Order::getStatusText($order['status']); ?></td>
$order['id']; ?>"
echo $order['id']; ?>"
<td><?php echo $order['address']; ?></td>
<td><?php echo $order['date']; ?></td>
<td><?php echo
<td>
    <a href="/cabinet/order/view/<?php echo
        title="Перегляд">
        <i class="fa fa-eye"></i>
    </a>
    <?php if ($order['status'] == '1'): ?>
        <a href="/cabinet/order/delete/<?php
            title="Видалити+*">
            <i class="fa fa-times"></i>
        </a>
    <?php endif; ?>
</td>
</tr>
<?php endforeach; ?>
</table>
</div>
</div>
</div>
<div class="tab-pane fade" id="account-details">
    <div class="login">
        <div class="container">
            <div class="row">
                <div class="col-sm-4 col-sm-4">
                    <?php if (isset($errors) && is_array($errors)):
?>
                        <ul>
                            <?php foreach ($errors as $error): ?>
                                <li> - <?php echo $error; ?></li>
                            <?php endforeach; ?>
                        </ul>
                    <?php endif; ?>
                    <div class="signup-form"><!--sign up form-->
                        <form action="#" method="post">
                            <div class="udasamnelabb">
                                <p>Ім'я <input id="usedabunkasdenum"
type="text" name="name"
                                placeholder="Ім'я"
                                value="<?php echo
$name; ?>"></p>
                                <p>Прізвище <input
                                name="lastname"
                                value="<?php echo
$id="usedabunkasdenum" type="text"
                                placeholder="Прізвище"
                                $lastname; ?>"></p>
                                <p>Телефон <input
                                name="phone"
                                value="<?php echo
$id="usedabunkasdenum" type="text"
                                placeholder="Телефон"
                                $phone; ?>"></p>
                                <p>Пошта <input id="usedabunkasdenum"
                                name="email"
                                value="<?php echo
type="text"
                                placeholder="Пошта"
                                $email; ?>" disabled/>
                                </p>

```

