МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК КУПІВЛІ ІНСТРУМЕНТІВ ТА ТОВАРІВ ДЛЯ САДІВНИЦТВА

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Максим КУШНІР**

«__» _____ 20__ р.

Керівник роботи

Канд. техн. наук, доцент, _____ **Євген ДАВИДЕНКО**

«__» _____ 20__ р.

Миколаїв – 2026

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«___» _____ 2026 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Кушніра Максима

1. Тема кваліфікаційної роботи «Вебзастосунок купівлі інструментів та товарів для садівництва» затверджена наказом ректора ЧНУ ім. Петра Могили № _349_ від «26» грудня 2025 р.

2. Строк представлення кваліфікаційної роботи «___» _____ 2026 р.

3. Очікуваний результат роботи та початкові дані якщо такі потрібні.

Очікуваним результатом роботи є розроблений вебзастосунок для купівлі інструментів та товарів для садівництва, що забезпечує можливість перегляду каталогу товарів, пошуку та фільтрації продукції, реєстрації та авторизації користувачів, формування кошика покупок, оформлення замовлення та адміністрування системи.

4. Перелік питань, що підлягають розробці:

– аналіз предметної області та існуючих програмних аналогів;

– визначення функціональних та нефункціональних вимог до системи;

– проєктування архітектури вебзастосунку;

– проєктування структури бази даних;

– розробка клієнтської частини вебзастосунку;

– розробка серверної частини системи;

– реалізація системи реєстрації та авторизації користувачів;

– реалізація функціоналу перегляду та пошуку товарів;

– реалізація системи кошика та оформлення замовлення;

– реалізація адміністративної панелі для управління товарами;

– тестування програмного забезпечення;

– аналіз результатів та формування висновків.

5. Перелік графічних матеріалів : Презентація

6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «____» _____ 2026 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Вебзастосунок купівлі інструментів та товарів для садівництва

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	26.12.2025	08.03.2026	виконано
2.	Огляд літератури за темою роботи	01.01.2026	25.03.2026	виконано
3.	Складання календарного плану КБР	26.02.2026	18.03.2026	виконано
4.	Аналіз предметної області	20.02.2026	10.03.2026	виконано
5.	Розробка проєктних рішень	05.03.2026	20.03.2026	виконано
6.	Моделювання та конструювання ПЗ	15.02.2026	30.02.2026	виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	01.03.2026	20.05.2026	виконано
8.	Відгук керівника КБР	22.05.2026	27.05.2026	виконано
9.	Оформлення КБР та презентації	13.05.2026	28.05.2026	виконано
10.	Попередній захист	28.05.2026	01.06.2026	виконано
11.	Рецензування	02.06.2026	08.06.2026	виконано
12.	Завершення оформлення КБР та презентації	08.06.2026	12.06.2026	виконано
13.	Захист кваліфікаційної роботи			

Здобувач _____ **Максим КУШНІР**

«__» _____ 20__ р.

Керівник роботи

Канд. техн. наук, доцент _____ **Євген ДАВИДЕНКО**

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

Вебзастосунок купівлі інструментів та товарів для садівництва

Здобувач 408 гр.: Кушнір Максим

Керівник: канд. техн. наук, доцент Давиденко Євген

У сучасних умовах стрімкого розвитку інформаційних технологій та електронної комерції все більшої популярності набувають вебзастосунки для здійснення онлайн–покупок. Значна кількість користувачів надає перевагу цифровим сервісам, які дозволяють швидко знайти необхідні товари, порівняти їх характеристики та оформити замовлення через Інтернет. Незважаючи на існування великих інтернет–магазинів, більшість з них є універсальними платформами і не орієнтовані на вузьку спеціалізацію у сфері садівництва. Тому актуальною є задача створення спеціалізованого вебзастосунку для продажу інструментів та товарів для садівництва, який забезпечить зручний пошук продукції, зрозумілу навігацію та ефективне управління каталогом товарів.

Об'єктом роботи є процес електронної комерції у сфері продажу інструментів та товарів для садівництва.

Предметом роботи є програмні засоби проєктування та реалізації вебзастосунків електронної комерції.

Метою роботи є розроблення вебзастосунку для купівлі інструментів та товарів для садівництва, який забезпечить зручний пошук продукції, перегляд характеристик товарів, оформлення замовлень через інтернет та ефективне управління каталогом товарів.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання.

У вступі обґрунтовано актуальність розробки вебзастосунку інтернет-магазину інструментів та товарів для садівництва, визначено мету, завдання, об'єкт та предмет дослідження.

У першому розділі проведено аналіз предметної області електронної комерції у сфері продажу інструментів та товарів для садівництва, розглянуто існуючі

аналоги інтернет–магазинів та сформовано специфікацію вимог до програмного забезпечення на основі аналізу їхніх переваг і недоліків.

Другий розділ присвячено моделюванню структури вебзастосунку, визначенню функціональних та нефункціональних вимог, а також опису ролей користувачів системи (клієнт, адміністратор, гість).

У третьому розділі описано архітектуру вебзастосунку, наведено UML–діаграми (діаграма класів, діаграма розгортання, діаграми використання), а також представлено структуру бази даних системи на основі SQL Server.

У четвертому розділі описано процес програмної реалізації вебзастосунку з використанням ASP.NET MVC, API, SQL Server, а також представлено результати тестування функціональності системи.

У висновках узагальнено результати виконаної роботи, підтверджено відповідність розробленого вебзастосунку поставленим вимогам та окреслено можливі напрями подальшого розвитку системи.

Кваліфікаційна робота викладена на 77 сторінках машинописного тексту, складається із вступу, 4 розділів, загальних висновків, переліку джерел посилання з 22 найменувань та 0 додатків. Праця містить 5 таблиць та 25 рисунків.

Ключові слова: вебзастосунок, електронна комерція, інтернет–магазин, ASP.NET MVC, API, SQL Server, онлайн–покупки, система управління товарами.

ABSTRACT

to the qualifying bachelor's thesis

Web application for purchasing gardening tools and products

Student of 408 group: Kushnir Maksym

Supervisor: PhD in Technical Sciences, Associate Professor Davydenko Yevhen

In today's rapidly developing information technology and e–commerce, web applications for online shopping are becoming increasingly popular. A significant number of users prefer digital services that allow them to quickly find the necessary products, compare their characteristics and place orders via the Internet. Despite the existence of large online stores, most of them are universal platforms and are not focused on a narrow specialization in the field of gardening. Therefore, the task of creating a specialized web application for selling tools and goods for gardening, which will provide convenient product search, clear navigation and effective management of the product catalog, is relevant.

The object of the work is the process of e–commerce in the field of selling tools and goods for gardening.

The subject of the work is software tools for designing and implementing e–commerce web applications.

The purpose of the work is to develop a web application for purchasing tools and goods for gardening, which will provide convenient product search, viewing product characteristics, placing orders via the Internet and effective management of the product catalog.

The qualification work consists of an introduction, 4 sections, conclusions and a list of references.

The introduction substantiates the relevance of developing a web application for an online store of gardening tools and goods, defines the goal, objectives, object and subject of the study.

The first section analyzes the subject area of e–commerce in the field of selling gardening tools and goods, considers existing analogues of online stores and forms a

specification of software requirements based on an analysis of their advantages and disadvantages.

The second section is devoted to modeling the structure of the web application, determining functional and non–functional requirements, as well as describing the roles of system users (client, administrator, guest).

The third section describes the architecture of the web application, provides UML diagrams (class diagram, deployment diagram, usage diagrams), and also presents the structure of the system database based on SQL Server.

The fourth section describes the process of software implementation of a web application using ASP.NET MVC, API, SQL Server, and also presents the results of testing the functionality of the system.

The conclusions summarize the results of the work performed, confirm the compliance of the developed web application with the requirements, and outline possible directions for further development of the system.

The qualification work is presented on 77 pages of typewritten text, consists of an introduction, 4 sections, general conclusions, a list of reference sources with 22 titles and 0 appendices. The work contains 5 tables and 25 figures.

Keywords: web application, e–commerce, online store, ASP.NET MVC, API, SQL Server, online shopping, product management system.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ... 3
ВСТУП ... 4
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ ... 6
1.1 Аналіз предметної області електронної комерції у сфері садівництва ... 6
1.2 Аналіз потреб користувачів інтернет–магазинів садівництва ... 8
1.3 Огляд та аналіз існуючих програмних аналогів ... 9
1.4 Постановка завдання ... 16
Висновки до розділу 1 ... 17
2 МОДЕЛЮВАННЯ ТА СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ... 19
2.1 Аналіз засобів та методів розробки ... 19
2.2 Моделі та методи ... 23
2.3 Моделювання функцій системи ... 24
2.4 Специфікація вимог до програмного забезпечення ... 26
Висновки до розділу 2 ... 32
3 АРХІТЕКТУРА, МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ... 34
3.1 Архітектура програмного забезпечення ... 34
3.2 Вибір технологій та компонент програмного забезпечення ... 36
3.3 UML–моделювання системи ... 37
3.4 Проєктування бази даних ... 47
3.5 Опис інтерфейсів застосунку ... 49
Висновки до розділу 3 ... 55
4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ... 56
4.1 Структура проєкту ... 56
4.2 Реалізація серверної частини ... 58
4.3 Реалізація клієнтської частини ... 62
4.4 Тестування вебзастосунку ... 63
4.5 Управління користувача ... 65
Висновки до розділу 4 ... 72
ВИСНОВКИ ... 74
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ... 76

ПЕРЕЛІК СКОРОЧЕНЬ

БД	–	База даних
ПЗ	–	Програмне забезпечення
.NET	–	Платформа розробки програмного забезпечення Microsoft
API	–	Application Programming Interface
ASP.NET	–	Active Server Pages .NET
CORS	–	Cross–Origin Resource Sharing
CRUD	–	Create, Read, Update, Delete
CSS	–	Cascading Style Sheets
HTML	–	HyperText Markup Language
HTTPS	–	HyperText Transfer Protocol Secure
JSON	–	JavaScript Object Notation
ORM	–	Object–Relational Mapping
REST	–	Representational State Transfer
SPA	–	Single Page Application
SQL	–	Structured Query Language
SSMS	–	SQL Server Management Studio
UI	–	User Interface
UML	–	Unified Modeling Language
URL	–	Uniform Resource Locator

ВСТУП

Ринок електронної комерції в Україні зростає навіть в умовах економічної нестабільності – і сегмент садівництва не є винятком. Покупці дедалі частіше шукають інструменти, насіння та добрива онлайн, а не в офлайн-магазинах. Особливо помітний цей зсув у категорії товарів для дому та присадибного господарства: попит на цифрові сервіси тут стабільно зростає протягом останніх кількох років. Сучасний користувач хоче швидко знайти потрібний товар, порівняти характеристики кількох моделей і оформити замовлення без зайвих кроків.

Однак більшість доступних платформ цю потребу закривають лише частково. Великі маркетплейси мають широкий асортимент, але перевантажений інтерфейс і відсутність садівничої спеціалізації. Вузькоспеціалізовані сайти орієнтовані на правильну аудиторію, але страждають від застарілого дизайну, відсутності мобільної версії та слабкої системи фільтрації. Жодна з доступних платформ не поєднує ці речі в одному рішенні.

У рамках даної кваліфікаційної роботи розроблено спеціалізований вебзастосунок GardenTools – інтернет-магазин інструментів та товарів для садівництва, орієнтований на три категорії користувачів: початківців, яким потрібні пояснення і рекомендації щодо вибору товару; досвідчених садівників, що цінують швидкий пошук і розвинену фільтрацію; та власників фермерських господарств із потребою у повторних замовленнях і зручному особистому кабінеті з історією покупок.

Науково-практичне значення роботи полягає в застосуванні сучасних методів проєктування вебсистем електронної комерції, використанні трирівневої архітектури програмного забезпечення, а також у створенні програмного продукту, що може бути розгорнутий як основа реального інтернет-магазину садових товарів малого або середнього бізнесу.

Метою кваліфікаційної роботи роботи є розроблення вебзастосунку для купівлі інструментів та товарів для садівництва, який забезпечить зручний пошук

продукції, перегляд характеристик товарів, оформлення замовлень через інтернет та ефективне управління каталогом товарів.

Для досягнення поставленої мети у кваліфікаційній роботі необхідно вирішити такі завдання:

1) провести аналіз предметної області та існуючих програмних аналогів;
2) визначити основні функціональні та нефункціональні вимоги до системи;
3) обрати архітектуру програмного забезпечення;
4) спроєктувати структуру бази даних системи;
5) розробити клієнтську частину вебзастосунку;
6) розробити серверну частину системи;
7) реалізувати систему реєстрації та авторизації користувачів;
8) реалізувати функціонал перегляду та пошуку товарів;
9) реалізувати систему кошика та оформлення замовлення;
10) реалізувати панель адміністратора для управління товарами;
11) провести тестування розробленого програмного забезпечення;
12) проаналізувати результати роботи та зробити відповідні висновки.

Об’єктом кваліфікаційної роботи є процес електронної комерції у сфері продажу інструментів та товарів для садівництва.

Предметом кваліфікаційної роботи є програмні засоби проєктування та реалізації вебзастосунків електронної комерції.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

У даному розділі проводиться системний аналіз предметної області електронної комерції у сфері продажу інструментів та товарів для садівництва. Розглядаються потреби цільової аудиторії, актуальні тенденції ринку онлайн–торгівлі, а також здійснюється огляд існуючих програмних аналогів. На підставі отриманих результатів формулюється постановка задачі кваліфікаційної роботи.

1.1 Аналіз предметної області електронної комерції у сфері садівництва

Електронна комерція посідає одне з провідних місць серед сегментів цифрової економіки за темпами зростання. За даними аналітичної компанії Statista, обсяг світового ринку електронної торгівлі у 2023 році перевищив 5,8 трлн доларів США, а до 2027 року прогнозується зростання до 7,9 трлн доларів [1]. Частка онлайн-покупок у загальному роздрібному обороті вже перевищила 20% і продовжує збільшуватися. В Україні ринок e-commerce також демонструє стійку позитивну динаміку: за оцінками EVO Group, кількість онлайн-замовлень щорічно зростає на 15–20%, а загальний обсяг ринку перевищив 150 млрд гривень на рік [2]. Рушіями цього зростання є підвищення цифрової грамотності населення, розширення покриття широкосмугового інтернету та розвиток інфраструктури онлайн-платежів і логістики.

Садівництво – одна з найпоширеніших сфер дозвілля і господарської діяльності в Україні. За даними Державної служби статистики України [3], понад 70% домогосподарств у сільській місцевості та близько 30% міських родин мають присадибні ділянки або дачі. Це формує стабільний попит на широку номенклатуру товарів: ручний садовий інструмент, механізоване обладнання, засоби захисту рослин, насіння, добрива, системи поливу. За оцінками аналітиків, ємність українського ринку садівництва перевищує 8 млрд гривень на рік, із яких частка онлайн-продажів становить близько 18% і має тенденцію до збільшення [3].

Організація онлайн-торгівлі в цьому сегменті має свою специфіку, яка суттєво відрізняє його від інших товарних категорій. Асортимент садових товарів охоплює

надзвичайно широкий ціновий діапазон – від дрібного ручного інвентарю вартістю кілька десятків гривень до великогабаритного обладнання ціною в десятки тисяч. Це потребує гнучкої системи каталогізації та фільтрації, де покупець може звузити вибір за кількома параметрами одночасно. Вибір конкретного товару нерідко залежить від типу ґрунту, площі ділянки та вирощуваних культур – тому детальні описи і практичні рекомендації щодо застосування є не додатковою опцією, а базовою вимогою до платформи. Окрема характеристика ринку – яскраво виражена сезонність: пік продажів припадає на березень–травень та серпень–вересень, тоді як у зимовий період попит падає на 60–70% відносно пікових значень. Така нерівномірність навантаження вимагає гнучкого керування асортиментом і складськими залишками.

Поведінка споживачів у цьому сегменті також має свої особливості. Дослідження Nielsen показують, що 64% покупців перед придбанням садового товару онлайн переглядають щонайменше три різні сайти для порівняння цін і характеристик [4]. При цьому 78% відмовляються від покупки, якщо сторінка завантажується довше трьох секунд, а 53% залишають сайт, якщо не знаходять потрібний товар протягом перших двох хвилин навігації [4]. Ці цифри прямо визначають пріоритети при проєктуванні: швидкодія і якість пошуку – не бажані характеристики, а обов'язкові.

Вимоги покупців у цьому сегменті можна розділити на дві групи. З боку контенту – детальні картки товарів із фотографіями і поясненням способу застосування, фільтрація за типом, призначенням і ціною, відгуки інших покупців і можливість порівняння моделей різних виробників. З боку процесу покупки – мінімальна кількість кроків при оформленні замовлення і адаптивний інтерфейс, що коректно працює на смартфоні. Останнє особливо актуально: значна частина аудиторії переглядає товари безпосередньо на ділянці. Власники інтернет-магазинів, своєю чергою, потребують автоматизованого керування каталогом, контролю складських залишків і аналітики продажів.

Наведені дані підтверджують: ринок онлайн-торгівлі садовими товарами в Україні достатньо великий і має специфічні вимоги, які більшість наявних платформ не закривають. Це і є практичним обґрунтуванням для розробки GardenTools..

1.2 Аналіз потреб користувачів інтернет-магазинів садівництва

Розробка вебзастосунку електронної комерції без розуміння цільової аудиторії – це проєктування наосліп. Перш ніж визначати функціональні вимоги до системи, необхідно з'ясувати, хто саме буде нею користуватися і чого від неї очікує. Користувачів інтернет-магазинів садівництва можна поділити на три групи залежно від рівня досвіду та характеру потреб.

Аматори-початківці – це люди, які лише починають займатися садівництвом. Вони приходять на сайт не лише за товаром, а й за інформацією: що взагалі купити, як це використовувати, що підійде для конкретної задачі. Для цієї групи навігація має бути інтуїтивно зрозумілою, картки товарів – містити детальний опис призначення і рекомендації щодо вибору, а інтерфейс не повинен перевантажувати зайвими опціями. Якщо людина витрачає більше двох хвилин на пошук і нічого не знаходить – вона йде.

Досвідчені садівники поводяться інакше. Вони точно знають, що шукають, і цінують насамперед швидкість: пошук за назвою або артикулом, розвинену фільтрацію за технічними характеристиками, можливість порівняти кілька моделей і почитати відгуки реальних покупців. Для них зайвий крок у навігації або відсутність потрібного фільтра – достатня причина перейти на інший сайт.

Власники фермерських господарств – це професійна аудиторія із зовсім іншим профілем потреб. Вони закуповують товари у великих обсягах, часто повторно, і потребують швидкого доступу до раніше замовлених позицій. Для них особистий кабінет з історією покупок – не зручна опція, а необхідність. Також важлива актуальна інформація про наявність товару на складі: замовляти те, чого немає, – витрата часу.

Аналіз типових сценаріїв поведінки в існуючих інтернет-магазинах садівництва дозволив виявити спільні потреби, характерні для всіх трьох груп. Швидкий і точний пошук товару за назвою, категорією або характеристикою – базова вимога незалежно від рівня досвіду користувача. Структура каталогу має бути логічною: інструменти, насіння, добрива, системи поливу – кожна категорія на своєму місці, без необхідності гортати десятки сторінок. Картки товарів повинні містити фотографії, технічні характеристики та опис застосування. Процес оформлення замовлення – мінімальна кількість кроків, без зайвих реєстраційних форм і перенаправлень. Особистий кабінет із відстеженням статусу замовлення і переглядом історії покупок закриває потреби як приватних покупців, так і фермерів. Адаптивний інтерфейс, що коректно відображається на смартфоні, – не побажання, а вимога: значна частина аудиторії переглядає і замовляє товари безпосередньо на присадибній ділянці.

Окремо варто розглянути потреби адміністратора системи. Це людина, яка відповідає за актуальність каталогу і обробку замовлень щодня. Їй потрібна зручна панель керування, де можна додати новий товар, відредагувати опис або ціну, видалити позицію, що знята з продажу, переглянути список замовлень і змінити їхні статуси. Якщо для оновлення ціни на лопату потрібно звертатися до розробника, система не придатна до реальної експлуатації.

Результати цього аналізу стали безпосередньою основою для формування функціональних вимог до GardenTools, детально описаних у розділі 2.

1.3 Огляд та аналіз існуючих програмних аналогів

Для визначення сильних і слабких сторін наявних рішень проаналізовано п'ять платформ, що здійснюють продаж товарів для садівництва в Україні: Rozetka, Агромаркет, Ідея-сад, Епіцентр та Coira..

ROZETKA (rozetka.com.ua) [5]

Rozetka є найбільшим українським маркетплейсом із аудиторією понад 20 млн унікальних відвідувачів на місяць і садовим асортиментом понад 15000

позицій. Пошук, фільтрація, відгуки, розгалужена мережа пунктів видачі і підтримка більшості платіжних сервісів – все реалізовано на рівні, достатньому для масової аудиторії. Є мобільний застосунок.

Але людина, що прийшла за конкретним садовим інструментом, стикається з інтерфейсом, перенасиченим рекламними блоками. Спеціалізованих фільтрів за агротехнічними характеристиками немає – пошук лопати нічим не відрізняється від пошуку смартфона. Досвідчений садівник, якому потрібно відфільтрувати інструмент за матеріалом держака або вагою, просто не знайде потрібного параметра і піде. Рекомендацій щодо застосування товарів платформа не надає. Навігація ускладнена надмірно розгалуженою ієрархією категорій, де садівництво губиться поміж тисячами інших підрозділів (рис. 1.1).

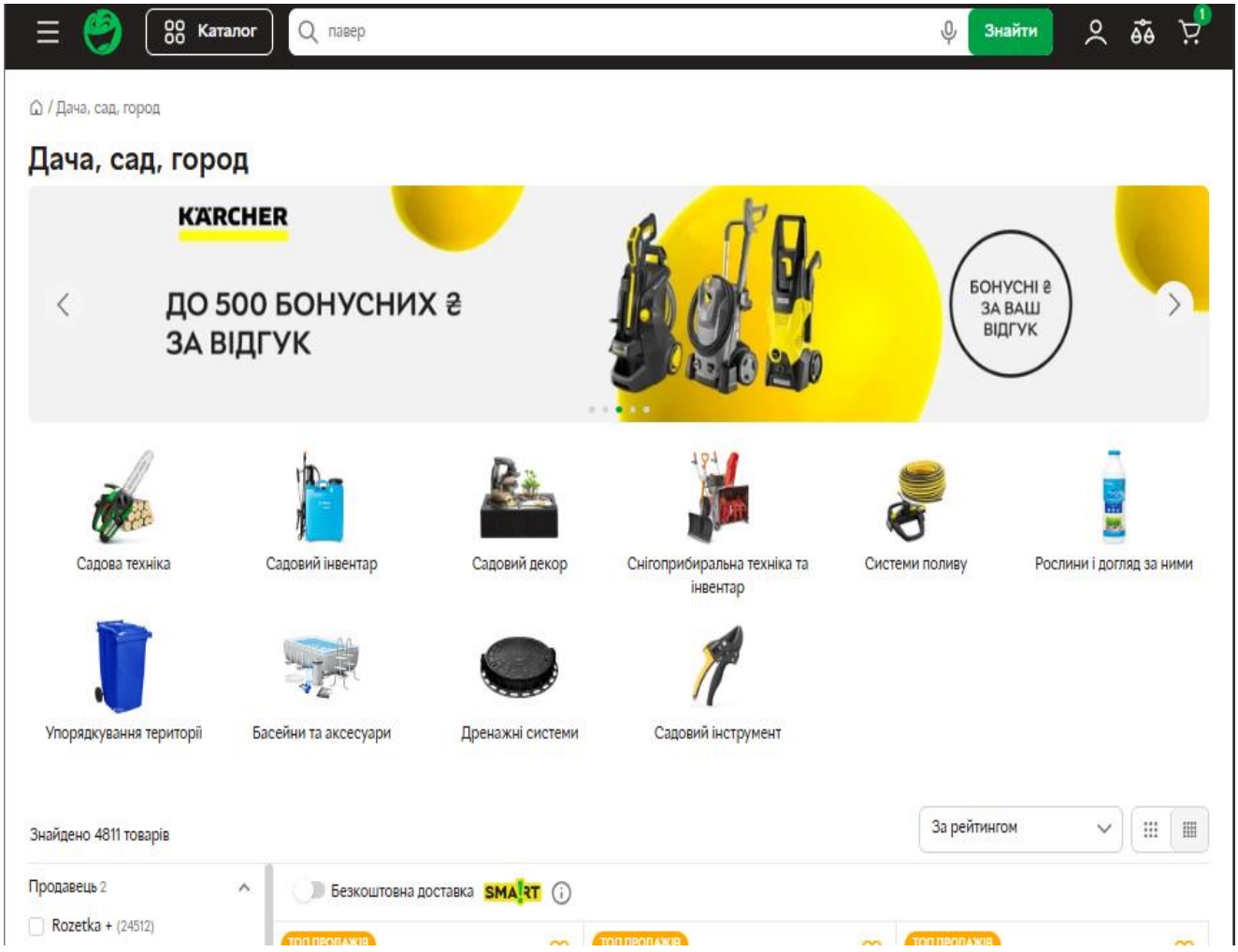

Рисунок 1.1 – Інтерфейс вебзастосунку ROZETKA

Агромаркет (agro–market.net/ua) [6]

Агромаркет орієнтований на фермерів і власників присадибних ділянок. Асортимент – насіння, добрива, засоби захисту рослин, інвентар. Картки товарів містять агрономічні описи з практичними рекомендаціями. Для фахової аудиторії це реальна перевага, якої немає у великих маркетплейсів, і саме це утримує частину покупців попри технічні недоліки платформи.

Дизайн не оновлювався роками. Швидкість завантаження низька. Адаптивної верстки немає – користувач зі смартфону бачить повну десктопну версію, якою на малому екрані неможливо нормально користуватися. Для садівничої аудиторії, що переглядає і замовляє товари просто на ділянці, це відсіює значну частину відвідувачів іще на етапі першого заходу на сайт. Фільтрація мінімальна, порівняти кілька позицій за характеристиками неможливо (рис. 1.2)..

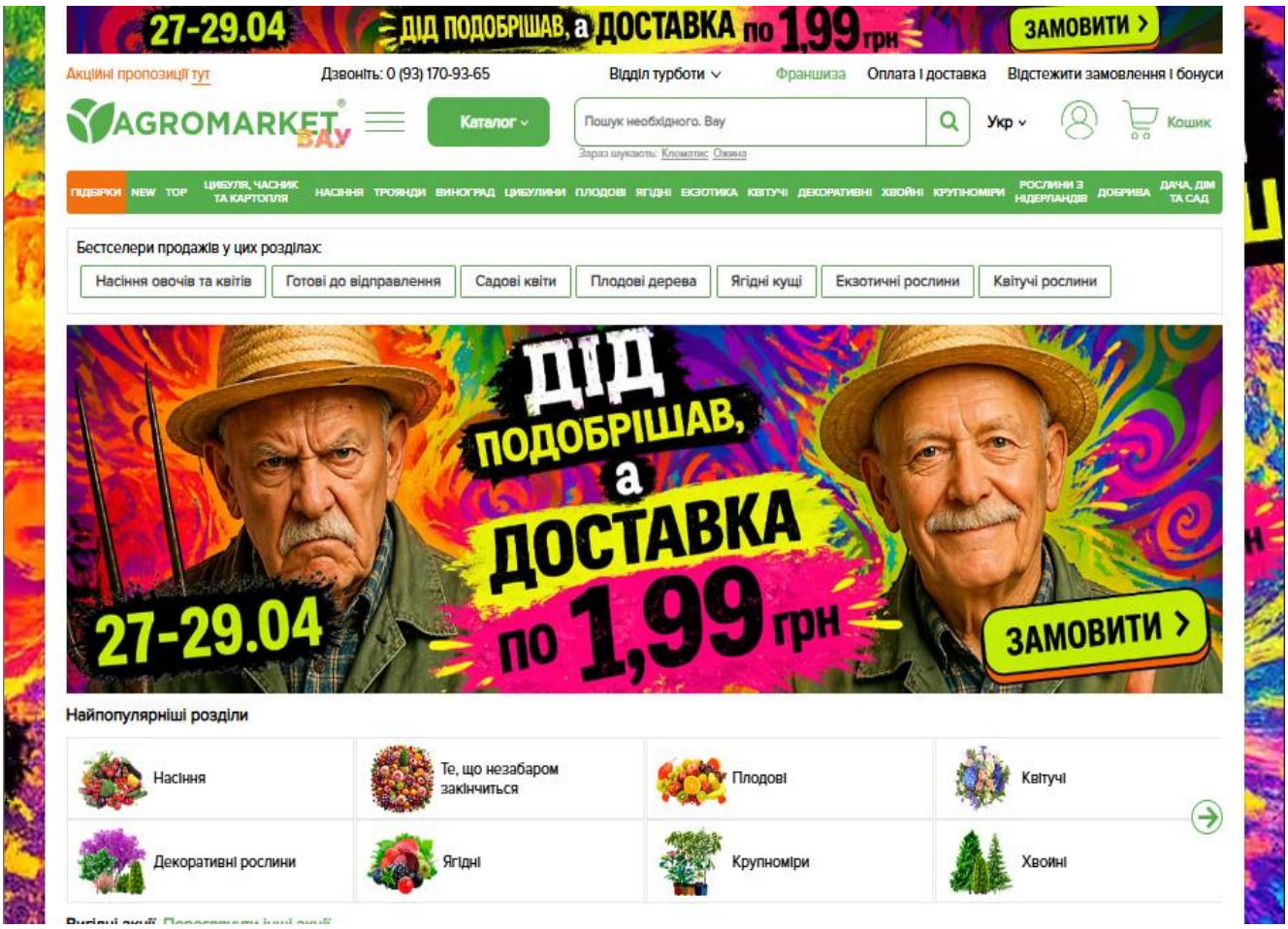

Рисунок 1.2 – Інтерфейс вебзастосунку Агромаркет

Ідея–сад (idea–sad.com.ua) [7]

З усіх п'яти аналогів Ідея-сад найближче підходить до концепції спеціалізованого садівничого магазину. Дизайн тематичний, каталог структурований логічно, є практичні статті для початківців. Мобільного застосунку немає.

Асортимент садового інструменту обмежений – користувач, який шукає щось конкретніше за базовий набір, швидко натрапляє на порожні категорії. Порівняння товарів відсутнє. Вибір між двома моделями секатора різних виробників покупець змушений робити вручну, відкриваючи по черзі дві вкладки. Особистого кабінету з історією замовлень немає: після оформлення покупки відстежити її статус через сайт неможливо. Для покупця, який замовив добрива до початку сезону і чекає доставки, це суттєва незручність. Фільтрація за технічними характеристиками не реалізована (рис. 1.3).

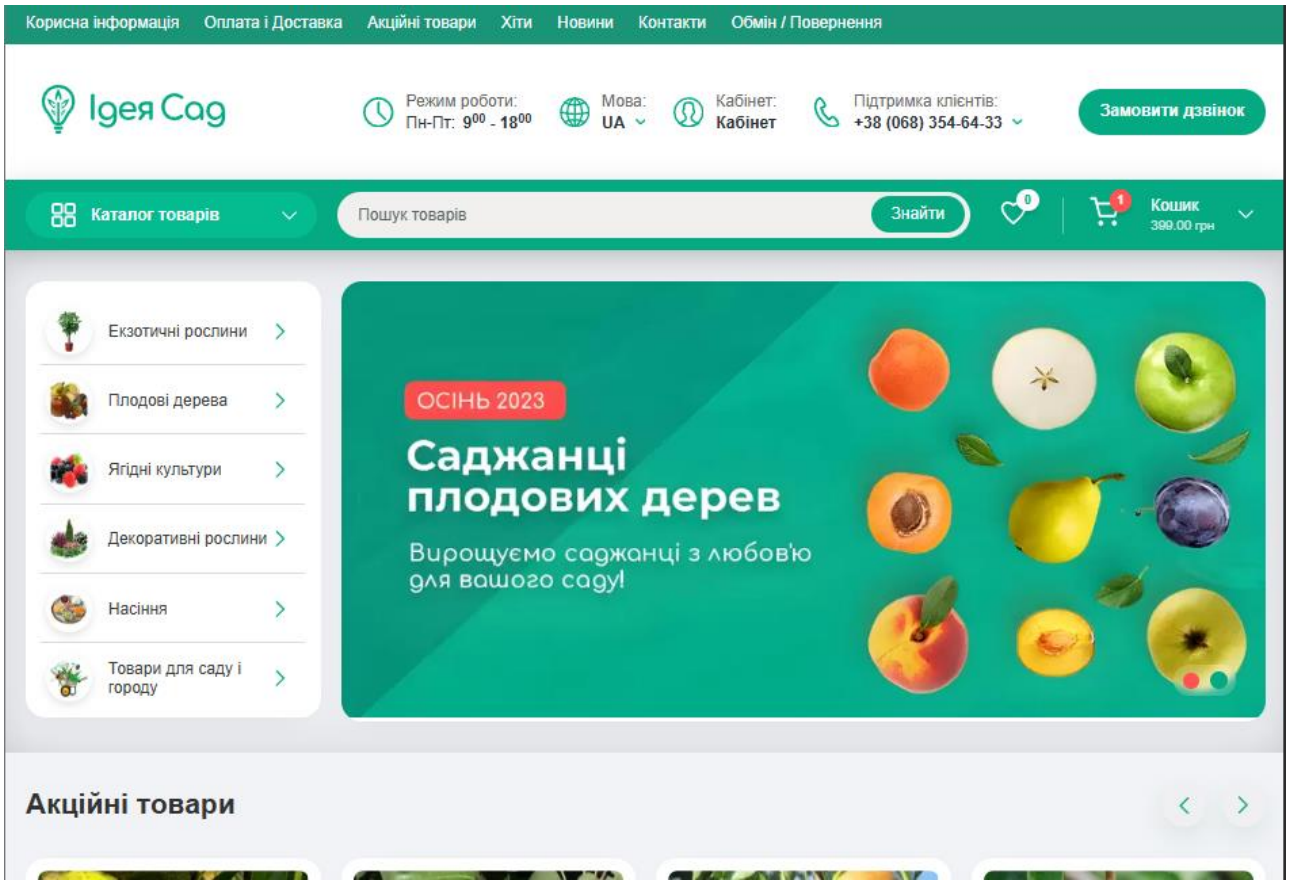

Рисунок 1.3 – Інтерфейс вебзастосунку Ідея–сад

Епіцентр (epicentrk.ua) [8]

Епіцентр – онлайн-магазин мережі будівельних гіпермаркетів. Садовий асортимент перевищує 10 000 позицій, є мобільний застосунок, самовивіз із фізичних магазинів по всій Україні і програма лояльності з накопиченням бонусів. Покупець, що цінує можливість подивитися товар наживо перед замовленням, тут знайде цю опцію.

Інтерфейс побудований під логіку гіпермаркету. Сотні категорій, складна навігація, жодного тематичного розділу для садівників – людина, що шукає систему поливу, проходить через кілька рівнів загальних категорій і нерідко губиться. Спеціалізованих рекомендацій щодо вибору садового інструменту немає. Початківець, якому потрібна порада щодо того, який культиватор підійде для конкретного типу ґрунту, не отримає на цій платформі жодної відповіді (рис. 1.4).

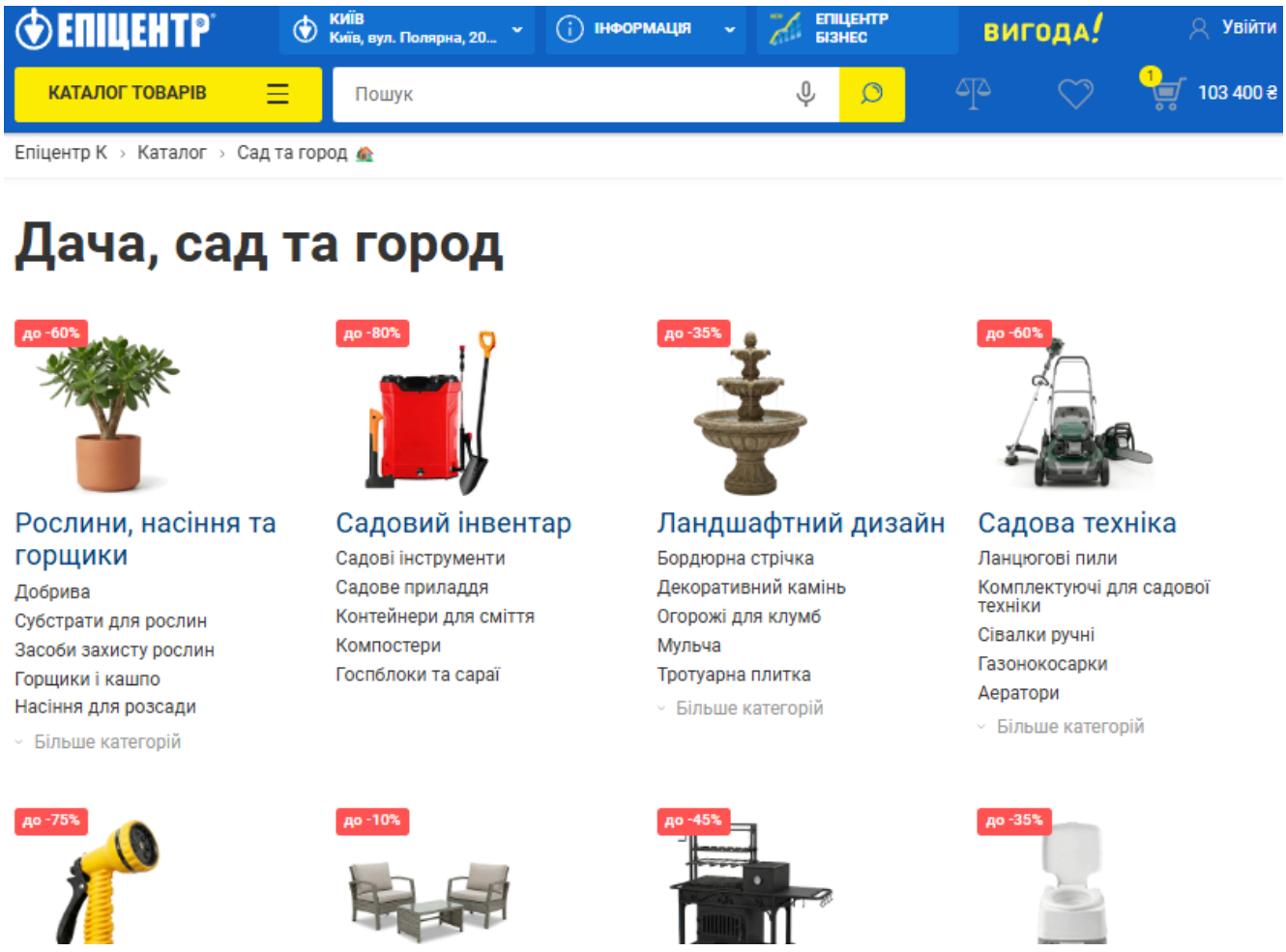

Рисунок 1.4 – Інтерфейс вебзастосунку Епіцентр

Coira (coira.com.ua) [9]

Coira – вузькоспеціалізований магазин для садівників, городників і власників дачних ділянок: насіння, добрива, інструменти, засоби захисту рослин. Детальні описи товарів із характеристиками і призначенням відрізняють платформу від маркетплейсів, де картка нерідко містить лише назву і ціну. Покупець, який знає що шукає, знайде тут необхідну технічну інформацію. Мобільного застосунку немає.

Дизайн застарілий. Адаптивного інтерфейсу немає – для аудиторії, що замовляє зі смартфону на ділянці, це відразу мінус. Порівняння товарів і система рекомендацій відсутні. Особистий кабінет є, але зводиться до перегляду поточного замовлення: повна історія покупок і швидке повторне замовлення недоступні (рис. 1.5).

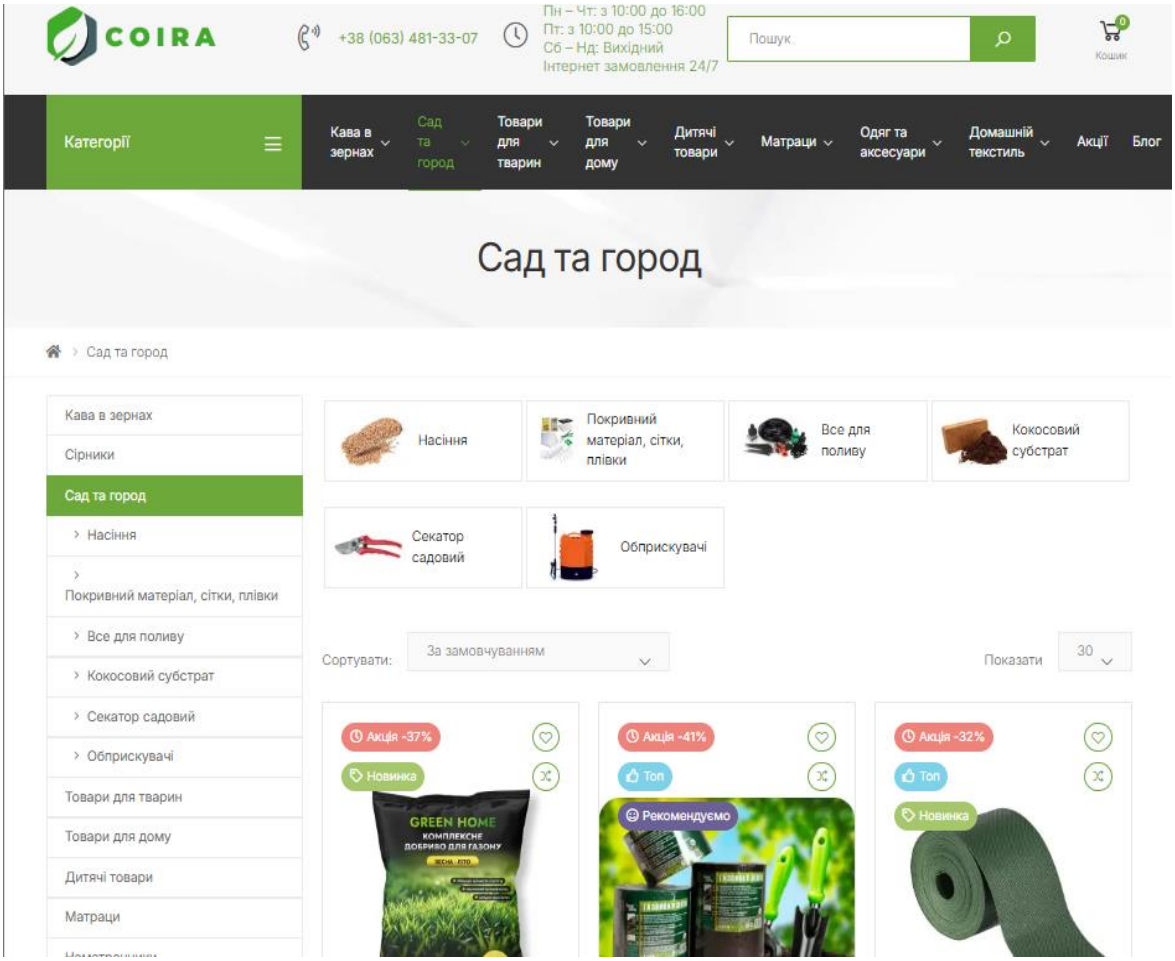

Рисунок 1.5 – Інтерфейс вебзастосунку Coira

Порівняльний аналіз

Для визначення сильних і слабких сторін складено порівняльну таблицю основних характеристик досліджених платформ на основі аналізу їхньої функціональності, зручності використання та рівня спеціалізації (табл. 1.1). Порівняння охоплює архітектурні рішення, технологічний стек, набір реалізованих функцій, а також сильні і слабкі сторони кожного рішення з погляду садівничої аудиторії.

Таблиця 1.1 – Порівняльний аналіз існуючих рішень

Назва	**Rozetka**	**Agro–Market**	**Idea–Sad**	**Epicentr**	**Coira**
Розробник (дистриб'ютор)	Rozetka Group	Agro–Market UA	Приватний інтернет-магазин	ТОВ «Епіцентр К»	Coira UA
Архітектура	3–tier web application (мікросервісні елементи)	Client–server web application	Client–server web application	3–tier web application	Client–server web application
Мова реалізації	Java, JavaScript (React)	PHP, JavaScript	PHP (CMS), JavaScript	Java, JavaScript (Vue.js)	PHP, JavaScript
Функції, характеристики	Каталог товарів, Розширений пошук і фільтри Кошик, Онлайн–оплата, Відгуки, Особистий кабінет	Каталог товарів, Пошук, Кошик, Оформлення замовлення, Опис агротехнічних характеристик	Каталог товарів, Кошик, Інформаційні статті, Пошук, Перегляд товарів	Каталог товарів, Пошук і фільтрація, Кошик, Самовивіз, Програма лояльності, Онлайн–оплата	Каталог товарів, Пошук, Кошик, Замовлення, Опис товарів
Переваги	Широкий асортимент; розвинена логістика; зручний інтерфейс; система відгуків; мобільна адаптація	Вузька спеціалізація; агрономічні описи; орієнтація на садівників; тематичний каталог	Тематичний дизайн; зрозуміла структура; наявність порад; орієнтація на садівництво	Великий асортимент; офлайн + онлайн; самовивіз; програма лояльності; надійність	Вузька спеціалізація; детальні описи; простий каталог

Кінець таблиці 1.1

Недоліки	Перевантажений інтерфейс; відсутність спеціалізованих фільтрів; складна навігація	Застарілий дизайн; низька швидкість; відсутність адаптивності; слабка фільтрація	Обмежений функціонал; відсутність особистого кабінету; немає порівняння товарів	Складна навігація; перевантаженість; слабка спеціалізація під садівництво	Застарілий дизайн; відсутність адаптивності; обмежений кабінет; немає рекомендацій

Аудит п'яти платформ виявив спільну архітектурну проблему ринку: жодна з досліджених систем не поєднує тематичну спеціалізацію на садівництві, сучасний адаптивний інтерфейс, розвинену систему фільтрації і повноцінний особистий кабінет в одному рішенні.

1.4 Постановка завдання

Проведений аналіз предметної області, тенденцій ринку та існуючих програмних аналогів дозволив сформулювати задачу кваліфікаційної роботи.

Мета розробки – створити спеціалізований вебзастосунок для купівлі інструментів та товарів для садівництва, що усуває виявлені недоліки існуючих рішень. Система має забезпечувати швидкий пошук товарів, їх детальний перегляд, оформлення замовлень онлайн та керування каталогом з боку адміністратора.

Функціональні вимоги розподілено на дві групи: з боку кінцевого користувача і з боку адміністратора.

З боку користувача система повинна забезпечувати:

– перегляд каталогу товарів із зручною навігацією по категоріях;

– пошук і фільтрацію товарів за назвою, категорією та ціновим діапазоном;

– перегляд детальної картки товару з описом, характеристиками та фотографіями;

– реєстрацію та авторизацію облікового запису;

– управління кошиком і оформлення замовлення з підтвердженням;

– доступ до особистого кабінету з переглядом історії та статусів замовлень;

– коректне відображення інтерфейсу на мобільних пристроях і десктопах.

З боку адміністратора система повинна забезпечувати:

– додавання, редагування та видалення товарів у каталозі;
– управління категоріями товарів;
– перегляд списку замовлень та зміну їхніх статусів;
– управління обліковими записами користувачів;
– контроль наявності товарів на складі.

Обидві групи вимог сформовані безпосередньо з результатів аналізу: кожна функція закриває конкретну прогалину, виявлену при огляді аналогів. Відсутність особистого кабінету в Ідея-саді і Coira – звідси вимога до історії замовлень. Відсутність мобільної адаптації в Агромаркеті і Coira – звідси вимога до адаптивного інтерфейсу. Слабка фільтрація у більшості платформ – звідси окрема вимога до пошуку за кількома параметрами одночасно.

GardenTools проєктується як застосунок, придатний до реального розгортання в малому або середньому бізнесі у сфері садівництва. Архітектурні рішення закладаються з урахуванням масштабованості: у майбутньому систему можна розширити системою рекомендацій, інтеграцією платіжних сервісів або програмою лояльності для постійних покупців без переробки ядра застосунку.

Висновки до розділу 1

Перший розділ присвячено аналізу предметної області електронної комерції у сфері продажу інструментів та товарів для садівництва. Ринок онлайн-торгівлі садовими товарами в Україні зростає стабільно і формує реальний попит на спеціалізовані цифрові сервіси. Специфіка сегменту визначається широким ціновим діапазоном асортименту, залежністю вибору товару від індивідуальних умов господарювання – типу ґрунту, площі ділянки, вирощуваних культур – та яскраво вираженою сезонністю попиту з піком у березні-травні та серпні-вересні.

Аналіз цільової аудиторії виявив три групи користувачів із різними поведінковими патернами: аматори-початківці, що потребують детальних описів і

рекомендацій щодо вибору; досвідчені садівники, орієнтовані на швидкий пошук і фільтрацію за технічними характеристиками; власники фермерських господарств, яким потрібні повторні замовлення і повноцінний особистий кабінет з історією покупок. Для кожної групи визначено пріоритетні функціональні очікування від платформи, що стало основою для формування вимог до системи.

Окремо досліджено тенденції розвитку ринку онлайн-торгівлі: зміщення попиту до спеціалізованих платформ, зростання частки мобільного трафіку, персоналізація клієнтського досвіду та спрощення процесу оформлення замовлень. Кожна з цих тенденцій знайшла відображення у конкретних функціональних вимогах до розроблюваного застосунку.

Огляд і порівняльний аналіз п'яти платформ – Rozetka, Агромаркет, Ідея-сад, Епіцентр та Coira – підтвердив системну проблему ринку: жодна з них не поєднує тематичну спеціалізацію на садівництві, сучасний адаптивний інтерфейс, розвинену систему фільтрації і повноцінний особистий кабінет в одному рішенні. Великі маркетплейси технічно розвинені, але побудовані під масову аудиторію без адаптації під садівничу нішу. Спеціалізовані платформи мають правильний предметний фокус, проте технічно застаріли і не підтримують мобільний доступ – що для аудиторії, яка переглядає і замовляє товари зі смартфону безпосередньо на ділянці, є суттєвим обмеженням.

За результатами проведеного аналізу сформульовано постановку задачі та визначено дві групи функціональних вимог до системи – з боку кінцевого користувача та з боку адміністратора. Їх повна деталізація у форматі специфікації вимог до програмного забезпечення наведена у розділі 2.

2 МОДЕЛЮВАННЯ ТА СПЕЦИФІКАЦІЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У даному розділі проводиться аналіз сучасного стану інструментарію, моделей та методів розробки вебзастосунків електронної комерції, визначаються ролі користувачів, моделюються функції системи та формується специфікація вимог до програмного забезпечення.

2.1 Аналіз засобів та методів розробки

Технологічний стек визначає продуктивність, масштабованість і зручність супроводу системи. Вибір інструментарію для GardenTools ґрунтується на аналізі практики розробки вебсистем електронної комерції та вимог, сформульованих у розділі 1.

Серверна частина

Платформа розробки – ASP.NET Core (.NET 8.0)

ASP.NET Core є кросплатформеною відкритою платформою від компанії Microsoft для побудови сучасних вебзастосунків і сервісів. У версії .NET 8.0, випущеній у 2023 році, платформа отримала суттєві покращення продуктивності. Дослідження Valiveti (2025) підтверджують переваги платформи у продуктивності й кросплатформеності при розробці сучасних вебзастосунків [10], зокрема на 35–40% вищу пропускну здатність порівняно з попередніми версіями ASP.NET Framework при обробці HTTP-запитів.

Для реалізації серверної частини обрано патерн MVC (Model-View-Controller), що є частиною ASP.NET Core і забезпечує чітке розмежування між рівнем представлення, бізнес-логікою та доступом до даних. Контролери виконують роль диспетчерів HTTP-запитів: вони приймають запит, викликають відповідний сервіс і передають результат у Razor View через ViewModel. Бізнес-логіка повністю винесена в окремі сервісні класи і не змішується з логікою контролера, що спрощує тестування окремих компонентів і подальшу підтримку

коду. Представлення формуються через Razor Views (.cshtml) – шаблони з підтримкою C#-виразів, що рендеряться на сервері і передаються клієнту у вигляді готового HTML.

Мова програмування – C#

C# є основною мовою розробки для платформи .NET. Сучасні версії мови – C# 12, що входить до .NET 8.0 – підтримують патерн-матчинг, nullable reference types, record-типи та інші можливості, що дозволяють писати безпечний і виразний код. Статична типізація мови зменшує кількість помилок ще на етапі компіляції, що особливо важливо при роботі з фінансовими розрахунками і бізнес-логікою замовлень.

Система управління базами даних – Microsoft SQL Server

Для зберігання даних обрано реляційну СКБД Microsoft SQL Server, яка забезпечує надійне зберігання структурованих даних, підтримку транзакцій, механізми резервного копіювання та розвинені засоби індексування. SQL Server органічно інтегрується з платформою .NET через Entity Framework Core – ORM-інструмент, що дозволяє працювати з базою даних в об'єктно-орієнтованому стилі без написання SQL-запитів вручну. Code-First підхід EF Core означає, що схема бази даних генерується з C#-моделей, а кожна структурна зміна фіксується у вигляді міграційного файлу, що робить розгортання передбачуваним і придатним для автоматизації. Детальний опис підходів до роботи з Entity Framework Core, зокрема реалізації CRUD-операцій та міграцій схеми бази даних, наведено у роботі Smith [11].

Клієнтська частина

HTML, CSS, JavaScript, Bootstrap, jQuery

Для побудови клієнтського інтерфейсу використовується поєднання стандартних вебтехнологій та бібліотек, що у сукупності забезпечують адаптивний і зручний інтерфейс користувача без залучення важких JavaScript-фреймворків.

HTML5 є основою розмітки всіх сторінок вебзастосунку. Використання семантичних елементів HTML5 – header, nav, main, section, article, footer – покращує структуру документа, спрощує підтримку коду та позитивно впливає на доступність застосунку для пошукових систем. CSS3 відповідає за візуальне оформлення сторінок: кольорову схему, типографіку, відступи, анімації та адаптивні правила відображення. Власні CSS-стилі доповнюють фреймворкові та дозволяють формувати зовнішній вигляд застосунку відповідно до тематики садівничого магазину.

Bootstrap 5 – провідний відкритий CSS-фреймворк, що надає готову бібліотеку адаптивних компонентів інтерфейсу: гнучку сітку на основі 12 колонок, кнопки, форми, навігаційні панелі, модальні вікна, картки товарів і спадні меню. Точки зламу (breakpoints) дозволяють автоматично адаптувати макет сторінки до різних розмірів екрану без написання власних медіа-запитів. Порівняльний аналіз Bootstrap з альтернативними CSS-фреймворками підтверджує його переваги за критеріями адаптивності та простоти використання [12]. Застосування Bootstrap 5 у даному проєкті прискорює розробку інтерфейсу та гарантує кросбраузерну сумісність компонентів.

JavaScript ES6+ є основною мовою програмування на стороні клієнта і використовується для реалізації динамічної поведінки інтерфейсу. Асинхронні запити до сервера виконуються через fetch API – зокрема для операцій з кошиком, де лічильник товарів у шапці сайту оновлюється без повного перезавантаження сторінки. Зміна кількості товарів у кошику реалізована через POST-форму зі стандартним перезавантаженням. Клієнтська валідація форм реєстрації та оформлення замовлення перевіряє коректність введених даних до їх відправки на сервер, що скорочує кількість зайвих серверних запитів і покращує досвід користувача. JavaScript також відповідає за відображення та приховування елементів інтерфейсу й обробку подій користувача.

jQuery присутній у проєкті як залежність Bootstrap і використовується переважно для клієнтської валідації форм. Центральна логіка динамічного інтерфейсу реалізована через нативний fetch API, а не через jQuery AJAX.

Середовище розробки – Visual Studio 2022

Як інтегроване середовище розробки використовується Microsoft Visual Studio 2022 – одне з найпотужніших середовищ розробки для платформи .NET. Visual Studio 2022 є 64-розрядним застосунком, що дозволяє ефективніше використовувати оперативну пам'ять при роботі з великими проєктами. Середовище надає повний набір інструментів: вбудований редактор Razor-шаблонів із підсвічуванням синтаксису та автодоповненням, відладчик із підтримкою точок зупину, інтегровану підтримку системи контролю версій Git, вбудований термінал, а також засоби профілювання продуктивності та аналізу покриття коду тестами. Це дозволяє вести розробку, налагодження і контроль версій в одному середовищі без перемикання між сторонніми інструментами.

Інструмент керування БД – Microsoft SQL Server Management Studio 22

Для адміністрування бази даних використовується Microsoft SQL Server Management Studio 22 (SSMS) – офіційний графічний інструмент від Microsoft для роботи з SQL Server. SSMS надає зручний інтерфейс для виконання SQL-запитів, перегляду та редагування структури таблиць, управління індексами, зовнішніми ключами та обмеженнями цілісності. У процесі розробки SSMS використовується для візуального проєктування схеми бази даних, перевірки коректності міграцій Entity Framework Core, налагодження запитів і контролю стану даних. Інструмент також підтримує резервне копіювання та відновлення бази даних, що є важливим на етапі тестування системи.

Вибір ASP.NET Core MVC зумовлений статичною типізацією C#, що зменшує кількість помилок на етапі компіляції, нативною інтеграцією з SQL Server та Entity Framework Core і підтвердженою продуктивністю платформи .NET 8.0. Цілісна екосистема Microsoft – від Visual Studio до SSMS – спрощує налаштування і усуває проблеми сумісності між компонентами. Легкий набір клієнтських технологій без

важких JavaScript-фреймворків забезпечує швидке завантаження сторінок, адаптивність інтерфейсу та мінімізує залежності проєкту, що спрощує його подальший супровід.

2.2 Моделі та методи

Проєктування вебзастосунків електронної комерції потребує формалізованих моделей і методів, що дозволяють структурувати вимоги, описати поведінку системи та обґрунтувати архітектурні рішення. На основі аналізу сучасних публікацій визначено методи та моделі, що застосовуються у даній роботі.

Вибір архітектурного стилю є рішенням, що визначає структуру, поведінку та властивості системи в цілому. Kim та Garlan [13] обґрунтовують важливість формального аналізу архітектурних стилів для забезпечення якості та супроводжуваності програмного забезпечення.

Архітектурний патерн – MVC (Model-View-Controller)

ASP.NET Core MVC реалізує патерн Model-View-Controller, де кожен запит обробляється контролером, який делегує бізнес-логіку сервісному шару, а результат передає у Razor View через ViewModel. Такий підхід забезпечує чітке розмежування між логікою обробки запитів і представленням даних. Gupta та ін. (2012) демонструють, що MVC-патерн є ефективним рішенням для вебзастосунків із чітко визначеною структурою сторінок і сервісною логікою, що характерно для інтернет-магазину [14]. Freeman та Sanderson підтверджують, що ASP.NET MVC забезпечує гнучку архітектуру з підтримкою тестування та розширення функціоналу без порушення існуючої структури проєкту [15].

Метод моделювання – UML

Для візуального моделювання структури та поведінки системи використовується Unified Modeling Language (UML) – стандартизована мова моделювання, що є загальновизнаним інструментом у програмній інженерії. У рамках даної роботи застосовуються такі типи UML-діаграм: діаграма варіантів використання (Use Case Diagram) – для опису взаємодії акторів із системою;

діаграма послідовностей (Sequence Diagram) – для моделювання міжкомпонентної взаємодії при виконанні ключових сценаріїв; діаграма діяльності (Activity Diagram) – для опису алгоритмів із розгалуженнями; діаграма класів (Class Diagram) – для опису структури доменної моделі і зв'язків між сутностями.

Метод проєктування бази даних – реляційна модель

Для проєктування структури зберігання даних застосовується реляційна модель на основі ER-діаграм (Entity-Relationship). Реляційний підхід забезпечує цілісність даних через механізм зовнішніх ключів, нормалізацію таблиць і транзакційну обробку. Реляційні СКБД залишаються стандартним рішенням для систем електронної комерції з передбачуваною структурою даних завдяки ACID-гарантіям, зрілості інструментарію та нативній інтеграції з ORM-бібліотеками [16, 17].

Метод специфікації вимог – SRS (Software Requirements Specification)

Для формалізації вимог до системи використовується підхід SRS відповідно до рекомендацій IEEE 830. Специфікація охоплює функціональні та нефункціональні вимоги, характеристики користувачів і системні обмеження, що детально викладені у підрозділі 2.4.

2.3 Моделювання функцій системи

Функціональна модель GardenTools побудована навколо трьох ролей користувачів: гість, клієнт та адміністратор. Кожна роль має визначений набір дозволів, що відповідає реальним сценаріям взаємодії з системою. Розмежування доступу між ролями реалізовано через ASP.NET Core Identity і контролюється на рівні контролерів застосунку.

Гість – незареєстрований відвідувач, що взаємодіє виключно з публічною частиною застосунку. Без авторизації йому доступний перегляд каталогу товарів і структури категорій, пошук та фільтрація за назвою, категорією і ціновим діапазоном, а також перегляд детальної картки обраного товару з повним описом і характеристиками. Перехід до форм реєстрації та входу в систему також відкритий

для гостя. Кошик, оформлення замовлення і особистий кабінет для незареєстрованого відвідувача недоступні – спроба перейти до цих розділів автоматично перенаправляє на сторінку входу.

Клієнт – зареєстрований і авторизований користувач, що отримує розширений набір функцій для здійснення покупок. Він має доступ до всього, що доступно гостю, і додатково може працювати з кошиком: додавати товари, коригувати їхню кількість через POST-форму зі стандартним оновленням сторінки, видаляти окремі позиції або очищати кошик повністю. Оформлення замовлення передбачає заповнення форми з контактними даними, адресою доставки і вибором способу оплати. Після підтвердження замовленню автоматично присвоюється початковий статус Pending і унікальний ідентифікатор. Особистий кабінет надає клієнту доступ до повної історії замовлень із датами, сумами і поточними статусами, а також до форми редагування персональних даних облікового запису.

Адміністратор – користувач із підвищеними правами доступу, що відповідає за операційне керування системою. Доступ до адміністративної панелі захищено атрибутом [Authorize(Roles = "Admin")] на рівні контролера, тому спроба неавторизованого входу одразу перенаправляє на сторінку входу. В межах панелі адміністратор виконує повний CRUD щодо товарів і категорій каталогу: створює нові позиції, редагує назви, описи, ціни та залишки, видаляє товари зі знятих з продажу категорій. Управління замовленнями реалізоване через перегляд повного списку з фільтрацією за статусом і послідовну зміну статусів по ланцюжку: Pending, Confirmed, Shipped, Delivered або Cancelled. Окремо адміністратор переглядає список зареєстрованих користувачів і може редагувати їхні облікові записи або обмежувати доступ.

Описаний розподіл ролей є архітектурною основою для побудови рольової авторизації і безпосередньо визначає структуру контролерів у проєкті: AccountController, AdminController, CartController, OrderController, ShopController і WishlistController – кожен обслуговує функціональний домен конкретної ролі або їхнього перетину.

2.4 Специфікація вимог до програмного забезпечення

Специфікація вимог визначає функціональну поведінку та ключові нефункціональні характеристики вебзастосунку для продажу товарів садівництва.

Призначення та межі проєкту

Вебзастосунок інтернет-магазину інструментів та товарів для садівництва забезпечує зручний процес вибору та придбання товарів через Інтернет. Покупцям надається доступ до каталогу з детальними характеристиками товарів, інструменти пошуку і фільтрації, кошик та особистий кабінет. Адміністраторам – інструменти управління каталогом, категоріями та замовленнями.

Погодження програмної документації

При розробці дотримуються таких стандартів і угод:

– ISO/IEC 25010:2011 – для визначення характеристик якості ПЗ;
– IEEE 830–1998 – для структури специфікації вимог;
– рекомендації OWASP – для забезпечення безпеки вебзастосунку.

Межі проєкту

До складу системи входять: вебінтерфейс для користувачів на основі ASP.NET Core MVC, Razor Views та контролерів, серверна бізнес–логіка, реляційна база даних та адміністративна панель.

Поза межами системи залишаються: реальна інтеграція з платіжними сервісами, модуль логістики та мобільний застосунок.

Загальний опис системи

Сфера застосування

Система призначена для малого та середнього бізнесу у сфері продажу садових товарів. Вебзастосунок може функціонувати як самостійна платформа онлайн–продажу або як доповнення до офлайн–магазину.

Характеристики користувачів

Характеристики користувачів системи були широко пояснені в розділі 2.3.

Загальна структура і склад системи

Система реалізована за трирівневою архітектурою:

– рівень представлення (Presentation Layer) – Razor Views (.cshtml), Bootstrap 5, JavaScript, адаптивний дизайн;

– рівень бізнес–логіки (Business Logic Layer) – контролери ASP.NET Core MVC, ViewModel-класи та сервіси обробки бізнес-правил, Entity Framework Core;

– рівень даних (Data Access Layer) – репозиторії для роботи з базою даних, міграції EF Core, Microsoft SQL Server.

Технологічні обмеження:

– система розроблена для роботи в середовищі .NET 8.0;

– мінімальна версія SQL Server – 2019;

– підтримка браузерів: Chrome 90+, Firefox 88+, Safari 14+, Edge 90+;

– мінімальна роздільна здатність екрану – 320 пікселів.

Юридичні обмеження:

– система має відповідати вимогам захисту персональних даних;

– використання захищеного з'єднання HTTPS для обробки чутливих даних.

Функції системи

Функція «Перегляд каталогу товарів»

Опис функції: система надає можливість перегляду каталогу товарів із фільтрацією та сортуванням.

Вхідна інформація: параметри фільтрації (категорія, мінімальна та максимальна ціна, назва); параметр сортування; номер сторінки.

Вихідна інформація: список товарів із назвою, зображенням і ціною; загальна кількість товарів; інформація про поточну сторінку пагінації.

Функціональні вимоги:

– FR–1.1: система відображає товари у вигляді сітки карток із назвою, фото та ціною;

– FR–1.2: система підтримує фільтрацію за категорією, ціновим діапазоном і назвою;

– FR–1.3: система підтримує пагінацію каталогу;

– FR–1.4: за відсутності товарів за заданими критеріями система відображає відповідне повідомлення.

Функція «Перегляд картки товару»

Опис: детальна інформація про обраний товар.

Вхідні дані: ідентифікатор товару.

Вихідні дані: назва, опис, характеристики, ціна, наявність, зображення.

Функціональні вимоги:

– FR–2.1: відображається повний опис і характеристики товару;

– FR–2.2: відображається залишок товару на складі;

– FR–2.3: наявна кнопка додавання до кошика.

Функція «Реєстрація та авторизація»

Опис: створення облікового запису та вхід до системи.

Вхідні дані: ім'я, електронна пошта, пароль.

Вихідні дані: авторизована сесія з відповідною роллю.

Функціональні вимоги:

– FR–3.1: перевіряється унікальність електронної пошти при реєстрації;

– FR–3.2: паролі зберігаються у хешованому вигляді;

– FR–3.3: доступ розмежовується відповідно до ролі користувача.

Функція «Управління кошиком»

Опис: формування переліку товарів перед оформленням замовлення.

Вхідні дані: ідентифікатор товару, кількість, тип дії.

Вихідні дані: вміст кошика із загальною вартістю.

Функціональні вимоги:

– FR–4.1: підтримується додавання, зміна кількості та видалення товарів;

– FR–4.2: загальна вартість перераховується автоматично;

– FR–4.3: вміст кошика зберігається між сеансами для авторизованих користувачів.

Функція «Оформлення замовлення»

Опис: створення замовлення на основі кошика з підтвердженням.

Вхідні дані: вміст кошика, контактні дані, адреса доставки.

Вихідні дані: підтвердження з унікальним номером замовлення.

Функціональні вимоги:

– FR–5.1: форма оформлення містить поля контактних даних та адреси доставки;

– FR–5.2: після підтвердження замовленню присвоюється статус «Нове» та унікальний номер;

– FR–5.3: після оформлення кошик очищається.

Функція «Особистий кабінет»

Опис: перегляд персональних даних та історії замовлень.

Вихідні дані: дані профілю, список замовлень зі статусами та датами.

Функціональні вимоги:

– FR–6.1: відображається перелік замовлень із датою, сумою та статусом;

– FR–6.2: надається можливість редагування персональних даних.

Функція «Адміністративна панель»

Опис: управління товарами, категоріями, замовленнями та користувачами.

Вхідні дані: дані товару або категорії; ідентифікатор і новий статус замовлення.

Функціональні вимоги:

– FR–7.1: повний CRUD для товарів і категорій каталогу;

– FR–7.2: перегляд і зміна статусів замовлень: **Нове** потім **В обробці** потім **Відправлено** потім **Доставлено / Скасовано**;

– FR–7.3: доступ до панелі обмежений виключно роллю Адміністратор.

Вимоги до інформаційного забезпечення

Джерела вхідної інформації

Вхідна інформація надходить з наступних джерел:

– дані товарів – вводяться адміністратором;

– дані користувачів – вводяться при реєстрації та редагуванні профілю;

– дані замовлень – формуються автоматично з форми доставки;

– зображення товарів – завантажуються за URL–посиланнями.

Нормативно–довідкова інформація

– довідник категорій товарів (назви та ідентифікатори);

– довідник статусів замовлень: Нове, В обробці, Відправлено, Доставлено, Скасовано;

– довідник ролей користувачів: Гість, Клієнт, Адміністратор.

Вимоги до організації та ведення інформації

– усі дані зберігаються у реляційній базі даних Microsoft SQL Server;

– схема бази даних керується міграціями Entity Framework Core;

– паролі зберігаються виключно у хешованому вигляді;

– цілісність зв'язків між таблицями забезпечується зовнішніми ключами;

– резервне копіювання бази даних має виконуватися регулярно.

Вимоги до технічного забезпечення

Вимоги до серверного обладнання:

– процесор: мінімум 2 ядра, рекомендовано 4 ядра;

– оперативна пам'ять: мінімум 4 ГБ, рекомендовано 8 ГБ;

– дисковий простір: мінімум 20 ГБ вільного місця;

– мережеве з'єднання: широкосмугове підключення до Інтернету.

Вимоги до клієнтського обладнання:

– будь–який пристрій із сучасним вебbraузером (ПК, ноутбук, смартфон);

– мінімальна роздільна здатність екрану: 320×568 пікселів;

– підключення до Інтернету зі швидкістю не менше 1 Мбіт/с.

Вимоги до програмного забезпечення

Архітектура: трирівнева MVC – контролери обробляють HTTP-запити, Razor Views формують представлення, сервіси містять бізнес-логіку, Entity Framework Core забезпечує доступ до даних.

Серверне ПЗ: .NET 8.0 Runtime, ASP.NET Core 8.0, Windows Server 2019+Мова і технології розробки

Мережне ПЗ: протокол HTTPS, cookies для сесій, JSON для AJAX–запитів.

ПЗ бази даних: Microsoft SQL Server 2019+, SSMS 22, Entity Framework Core 8.0.

Мови та технології:

backend – C# 12, .NET 8, ASP.NET Core MVC, ASP.NET Core Identity, EF Core 8;

frontend – HTML5, CSS3, Bootstrap 5, JavaScript ES6+, fetch API, jQuery validation;

інструменти – Visual Studio 2022, SSMS 22.

Вимоги до зовнішніх інтерфейсів

Інтерфейс користувача

– адаптивний дизайн від 320 px до 1920 px на основі Bootstrap 5;

– інтуїтивна навігація: головна, каталог, картка товару, кошик, особистий кабінет;

– уніфікований дизайн на всіх сторінках системи;

– клієнтська валідація форм перед відправкою на сервер.

Апаратний інтерфейс:

Спеціалізоване апаратне забезпечення не потрібне. Система функціонує на будь–якому пристрої з підтримкою сучасного веббраузера.

Програмний інтерфейс

Взаємодія між клієнтською та серверною частиною здійснюється через серверний рендеринг Razor Views та окремі асинхронні запити через fetch API для динамічного оновлення вмісту кошика.

Комунікаційний протокол

– HTTPS для всіх запитів;
– HTTP–методи GET та POST для взаємодії з MVC-контролерами;
– підтримка cookies для автентифікації та збереження сесії.

Властивості програмного забезпечення

Доступність – готовність системи не менше 99%, цілодобовий доступ.

Супроводжуваність – модульна структура MVC: контролери, сервіси, ViewModel та Razor Views, документування методів, логування подій.

Переносимість – розгортання на Windows та Linux без прив'язки до апаратного забезпечення.

Продуктивність – відповідь сервера до 2 секунд, завантаження сторінки до 3 секунд, підтримка 100+ одночасних користувачів.

Надійність – транзакційність операцій через EF Core, серверна та клієнтська валідація, коректна обробка помилок.

Безпека – хешування паролів, захист від SQL–ін'єкцій через EF Core, захист від XSS, CSRF-токени у формах Razor Views через ASP.NET Core MVC, розмежування доступу за ролями, обов'язковий HTTPS.

Висновки до розділу 2

У другому розділі визначено технологічну, методичну та вимогову основу вебзастосунку GardenTools. Для серверної частини обрано ASP.NET Core MVC на платформі .NET 8.0, мову C#, Entity Framework Core та Microsoft SQL Server. Обраний стек відповідає структурі проєкту: контролери приймають HTTP-запити, сервіси містять бізнес-логіку, а Razor Views разом із ViewModel формують сторінки для користувача.

Клієнтська частина побудована на HTML5, CSS3, Bootstrap 5, JavaScript ES6+, fetch API та jQuery validation. Bootstrap 5 використовується для адаптивної верстки й готових компонентів інтерфейсу. JavaScript відповідає за окремі

динамічні дії, зокрема асинхронне оновлення окремих елементів сторінки, а jQuery застосовується переважно для клієнтської валідації форм.

У розділі також визначено методи проєктування системи. Для архітектури застосовано MVC, для опису структури й поведінки системи використано UML-діаграми, для бази даних обрано реляційну модель, а для формалізації вимог використано підхід SRS відповідно до IEEE 830–1998. Це дозволило описати не тільки функції застосунку, а й обмеження, інтерфейси та вимоги до якості програмного забезпечення.

Функціональна модель GardenTools побудована навколо ролей гостя, клієнта та адміністратора. Гість має доступ до відкритої частини сайту, клієнт після авторизації працює з кошиком, замовленнями й особистим кабінетом, а адміністратор керує каталогом, категоріями, замовленнями та користувачами. Розмежування доступу реалізується через ASP.NET Core Identity.

Сформована специфікація охоплює перегляд каталогу, картку товару, реєстрацію та авторизацію, управління кошиком, оформлення замовлення, особистий кабінет і адміністративну панель. Окремо описано інформаційне, технічне та програмне забезпечення, зовнішні інтерфейси, вимоги до продуктивності, надійності й безпеки. Матеріали цього розділу використовуються далі для проєктування архітектури, структури бази даних і програмної реалізації вебзастосунку.

3 АРХІТЕКТУРА, МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У цьому розділі описано проєктні рішення, на яких побудовано вебзастосунок GardenTools. До них належать архітектура програмного забезпечення, UML-моделі, структура реляційної бази даних і користувацькі інтерфейси. Ці рішення використовуються далі під час опису програмної реалізації у розділі 4.

3.1 Архітектура програмного забезпечення

GardenTools реалізовано за трирівневою архітектурою (3-tier architecture). У такій структурі кожен шар має власну зону відповідальності і взаємодіє переважно із суміжними частинами системи. Це спрощує супровід проєкту, бо зміни в одному шарі не потребують повної перебудови всієї системи.

Рівень представлення (Presentation Layer) охоплює Razor Views (.cshtml) і статичні ресурси в директорії wwwroot: CSS-файли Bootstrap 5, власні стилі та JavaScript-скрипти. Сторінки застосунку прив’язані до відповідних Controller/Action-маршрутів і рендеряться на сервері. JavaScript використовується точково: для роботи з елементами інтерфейсу, клієнтської валідації та окремих асинхронних запитів через fetch API, зокрема для оновлення лічильника кошика у шапці сайту без перезавантаження сторінки.

Рівень бізнес-логіки (Business Logic Layer) зосереджений у директорії Services. Сервісні класи не працюють із HTTP-контекстом напряму, а оперують доменними об’єктами з папки Models і отримують дані через AppDbContext, який передається механізмом Dependency Injection. До цього шару належать операції перерахунку суми кошика, перевірки залишків товару на складі, розрахунку знижок і оформлення замовлення. Такий поділ дає змогу перевіряти логіку сервісів окремо від контролерів.

Рівень даних (Data Access Layer) реалізований через Entity Framework Core 8.0. AppDbContext є основною точкою доступу до Microsoft SQL Server, а міграції EF Core фіксують зміни структури бази даних у вигляді окремих файлів. Клас SeedData використовується для початкового наповнення бази довідковими даними та створення стартового адміністративного користувача.

Контролери AccountController, AdminController, CartController, HomeController, OrderController, ShopController і WishlistController виконують роль диспетчерів HTTP-запитів. Вони приймають запит, викликають відповідний сервіс і повертають результат у вигляді Razor View або відповіді іншого типу. Бізнес-правила не зосереджуються в контролерах, що відповідає принципу єдиної відповідальності (SRP) з SOLID.

Залежності між компонентами керуються стандартним IoC-контейнером ASP.NET Core. Реєстрація сервісів, AppDbContext, ASP.NET Core Identity та middleware виконується у файлі Program.cs. Окремо у рішенні наявний модуль GardenTools.Tests, який використовується для перевірки сервісного шару.

Таблиця 3.1 – Розподіл компонентів за архітектурними шарами

Шар	**Директорія / компонент**	**Відповідальність**
Presentation	Views/, wwwroot/	Razor–шаблони, CSS, JS, Bootstrap 5
Business Logic	Controllers/, Services/, ViewModels/	Обробка HTTP, бізнес–правила, DTO
Data Access	Data/ (AppDbContext, SeedData), Migrations/	ORM–доступ до SQL Server, міграції схеми
Domain	Models/, DTOs/, Extensions/	Доменні сутності, трансформація даних

Розподіл компонентів за шарами показує, що проєкт не зводиться лише до набору контролерів і представлень. Основна логіка винесена в сервіси, а доменні сутності та доступ до даних відокремлені від інтерфейсу користувача.

3.2 Вибір технологій та компонент програмного забезпечення

Технологічний стек підібрано з урахуванням продуктивності, сумісності компонентів і кількості зовнішніх залежностей. Детальне обґрунтування вибору наведено у розділі 2, тому в цьому підрозділі подано стислий опис ролі кожної технології в архітектурі застосунку.

ASP.NET Core 8.0 з патерном MVC використовується як основа серверної частини. Вбудований Kestrel-сервер обробляє HTTP-запити, а middleware-конвеєр підключає Cookie-аутентифікацію через ASP.NET Core Identity, авторизацію за ролями, маршрутизацію та роздачу статичних файлів із wwwroot. ASP.NET Core Identity відповідає за реєстрацію, вхід, ролі користувачів і хешування паролів через вбудований PasswordHasher. Дані Identity зберігаються у таблицях AspNetUsers, AspNetRoles та пов’язаних таблицях бази даних.

Entity Framework Core 8.0 використовується як ORM-рівень між C#-кодом і Microsoft SQL Server. Через AppDbContext застосунок виконує LINQ-запити, які EF Core перетворює у SQL-команди. Підхід Code-First дозволяє описувати структуру даних у C#-моделях, а зміни схеми переносити до бази через міграції без ручного написання DDL-скриптів.

Bootstrap 5 використовується для побудови адаптивного інтерфейсу. Його grid-система на 12 колонок і точки зламу xs/sm/md/lg/xl дають змогу налаштувати відображення сторінок для екранів від 320 до 1920 пікселів. HTML5 і CSS3 відповідають за структуру та стилізацію сторінок, а власні CSS-файли задають візуальний стиль GardenTools.

JavaScript ES6+ застосовується для окремих динамічних сценаріїв на клієнтській стороні. Асинхронні запити до сервера виконуються через fetch API. jQuery 3.7 у проєкті використовується переважно разом із jquery.validate та jquery.validate.unobtrusive для клієнтської валідації форм. Такий підхід дозволяє не підключати важкі JavaScript-фреймворки для задач, які в межах цього застосунку вирішуються простішими засобами.

Таблиця 3.2 – Перелік технологій та їхнє призначення

Технологія / бібліотека	Версія	Призначення в GardenTools
ASP.NET Core MVC	8.0	HTTP–обробка, маршрутизація, рендеринг Razor–шаблонів
ASP.NET Core Identity	8.0	Реєстрація, вхід, ролі, хешування паролів
Entity Framework Core	8.0	ORM, міграції, Code–First схема бази даних
Microsoft SQL Server	2019+	Реляційна СКБД, зберігання всіх даних системи
Bootstrap	5.3	Адаптивна сітка, UI–компоненти, кросбраузерна сумісність
jQuery	3.7	AJAX–запити кошика, DOM–маніпуляції
C#	12 (.NET 8)	Основна мова серверної розробки
HTML5 / CSS3	–	Розмітка та стилізація клієнтської частини

Наведений набір технологій узгоджується з архітектурою MVC і не потребує складного клієнтського фреймворку. Для GardenTools достатньо серверного рендерингу Razor Views, адаптивної верстки Bootstrap і точкового використання JavaScript для динамічних дій.

3.3 UML–моделювання системи

Поведінку та структуру GardenTools описано за допомогою UML-діаграм. У роботі використано діаграму варіантів використання, діаграму класів, діаграми послідовностей і діаграми діяльності. Разом вони показують, як користувачі взаємодіють із системою, які сутності входять до доменної моделі та як виконуються основні сценарії на рівні контролерів, сервісів і бази даних.

Діаграма варіантів використання

Діаграма варіантів використання (рис. 3.1) описує взаємодію системи з трьома акторами: Гість, Клієнт та Адміністратор. У цій діаграмі не деталізується

програмна реалізація. Вона фіксує набір функцій, доступних кожній ролі, і показує межі відповідальності системи з погляду користувача.

Гість є відвідувачем без облікового запису. Йому доступні перегляд каталогу товарів, відкриття картки конкретного товару, пошук, фільтрація та перехід до форми реєстрації або входу. Персоналізовані функції для гостя обмежені, оскільки кошик, оформлення замовлення, список бажань і особистий кабінет потребують ідентифікації користувача. Після авторизації користувач переходить до ролі Клієнта.

Клієнт має доступ до функцій гостя і додатково працює з покупками. Основний сценарій для цієї ролі пов’язаний з оформленням замовлення: користувач додає товари до кошика, змінює кількість позицій, видаляє непотрібні товари, після чого переходить до форми оформлення. Для створення замовлення необхідно вказати адресу доставки та спосіб оплати. Клієнт також переглядає історію замовлень, користується списком бажань і нотатками садового планувальника. Окремий сценарій пов’язаний із заявкою на сповіщення про наявність товару, якщо потрібна позиція тимчасово відсутня на складі.

Адміністратор працює з операційною частиною системи. Йому доступне управління товарами, категоріями, брендами, знижками та сумісними товарами. Окремий блок функцій стосується замовлень: адміністратор переглядає список замовлень, відкриває їхні деталі та змінює статуси. Також він керує користувачами, модерує відгуки, переглядає заявки StockNotifications, використовує сторінки звітів і аналітики. Через це адміністративна частина діаграми має більше варіантів використання, ніж клієнтська.

Загалом діаграма містить близько 25 варіантів використання. Найбільша їх частина належить адміністратору, оскільки управління каталогом і замовленнями охоплює більше дій, ніж звичайний сценарій покупки. Клієнтська частина при цьому зберігає повний шлях користувача: від перегляду товарів до оформлення замовлення і подальшого перегляду історії покупок.

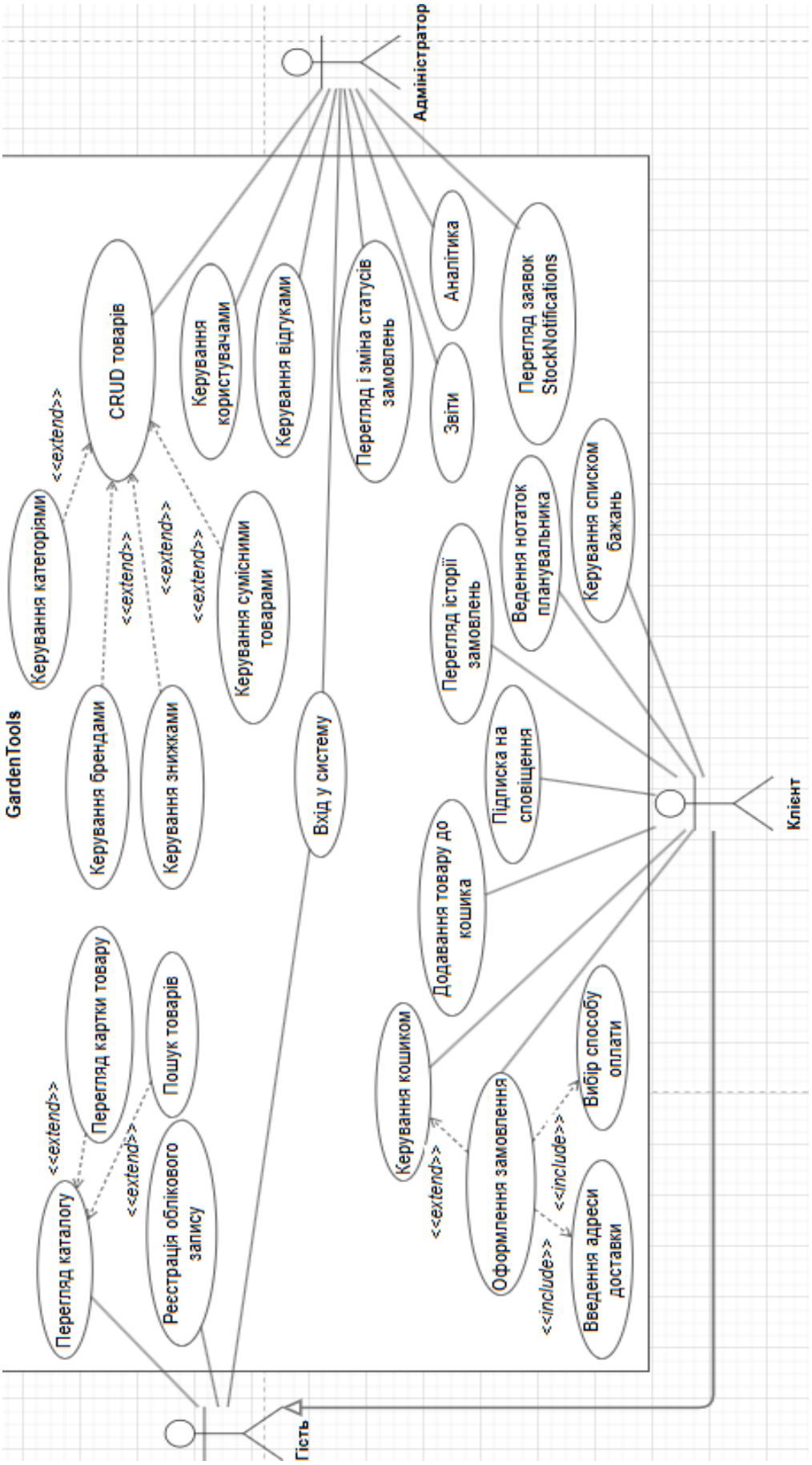

Рисунок 3.1 – Діаграма варіантів використання GardenTools

Діаграма класів

Діаграма класів (рис. 3.2) описує доменну модель GardenTools. Центральною сутністю каталогу є клас Product. Він містить назву, опис, ціну, залишок на складі, URL зображення, інструкцію використання, технічні характеристики та прапор HasCalculator. Product пов’язаний із довідниками Category, Brand і Season, а також із сутностями Discount, Review, ProductBundle і CartItem.

Клас ApplicationUser успадковує IdentityUser і розширює стандартну модель користувача полями FirstName, LastName, Address і RegisteredAt. Через ідентифікатор користувача з ним пов’язані Order, CartItem, WishlistItem, Review, PlannerNote і UserAddress. UserAddress дозволяє зберігати кілька адрес доставки для одного користувача. Одна з них може бути позначена як основна за допомогою прапора IsDefault.

Зв’язок між Order і OrderItem має характер композиції: позиції замовлення існують у межах конкретного замовлення і не мають самостійного значення без нього. У OrderItem зберігається UnitPrice, тобто ціна товару на момент оформлення. Це потрібно, щоб подальша зміна ціни у каталозі не змінювала вже створене замовлення. Статус замовлення і спосіб оплати винесено в окремі переліки: OrderStatus (Pending, Confirmed, Shipped, Delivered, Cancelled) і PaymentMethod (Card, Cash).

ProductBundle використовується для опису сумісних товарів. Для цього в моделі застосовано два зовнішні ключі на Product: MainProductId і RelatedProductId. Discount зберігає відсоток знижки, дати її дії та прапор IsActive. StockNotification фіксує заявки користувачів на повідомлення про повернення товару в наявність. PlannerNote зберігає нотатки садового планувальника і може мати прив’язку до товару через необов’язкові поля.

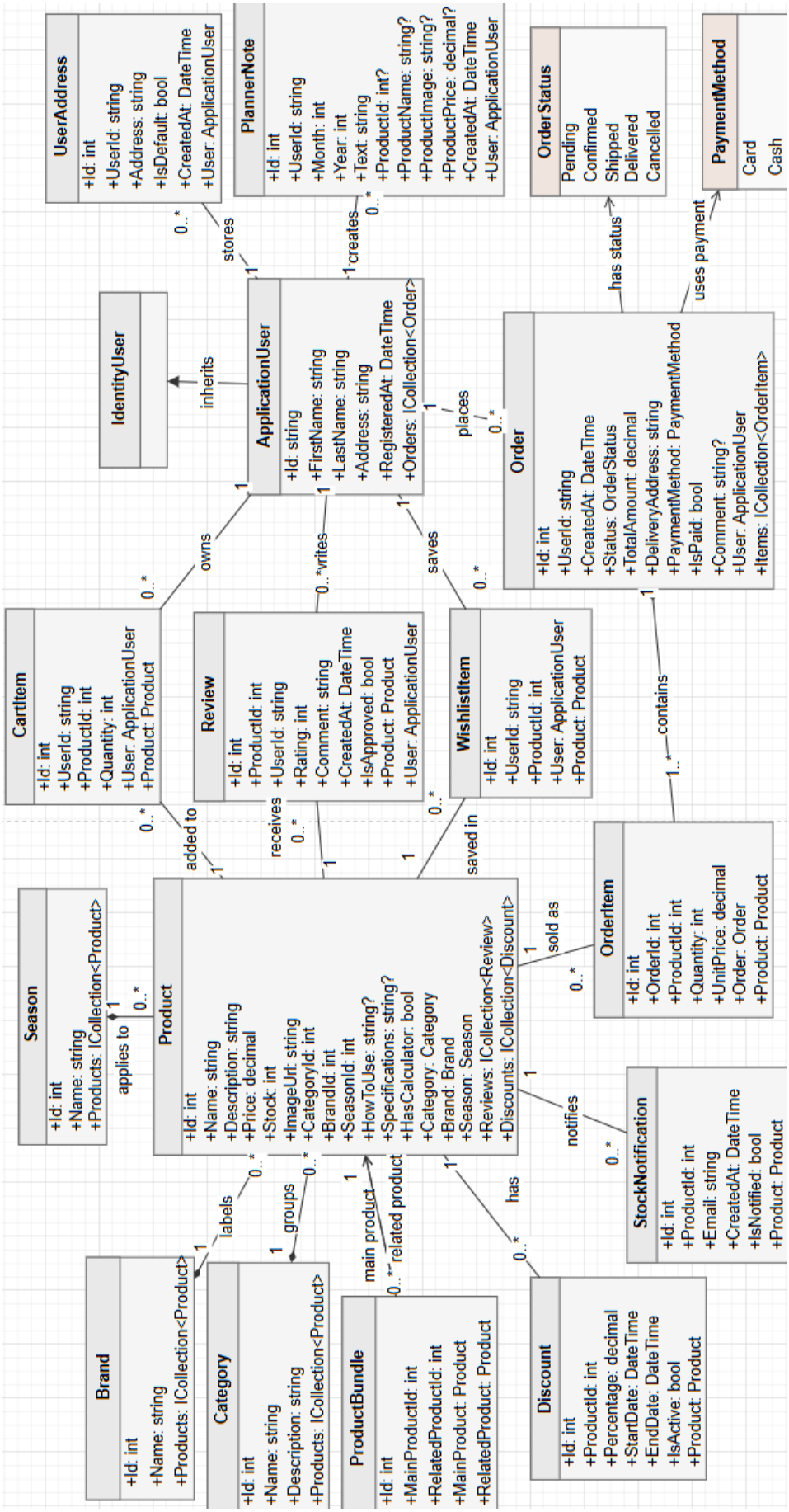

Рисунок 3.2 – Діаграма класів основних сутностей GardenTools

Діаграми послідовностей

Діаграми послідовностей (Sequence Diagrams) показують порядок викликів між браузером, контролером, сервісним шаром, AppDbContext і SQL Server. Для GardenTools розглянуто сценарії реєстрації користувача, додавання товару до кошика та оформлення замовлення.

Реєстрація користувача починається з POST-запиту на AccountController.Register. Користувач заповнює форму, після чого контролер перевіряє модель і передає дані до UserManager.CreateAsync. ASP.NET Core Identity створює запис у таблиці AspNetUsers і хешує пароль за допомогою вбудованого PasswordHasher. Після успішної реєстрації користувачу призначається роль User, виконується автоматичний вхід через SignInManager.SignInAsync і відбувається перенаправлення на головну сторінку. Якщо електронна пошта вже використовується або дані не проходять перевірку, форма відкривається повторно з повідомленням про помилку.

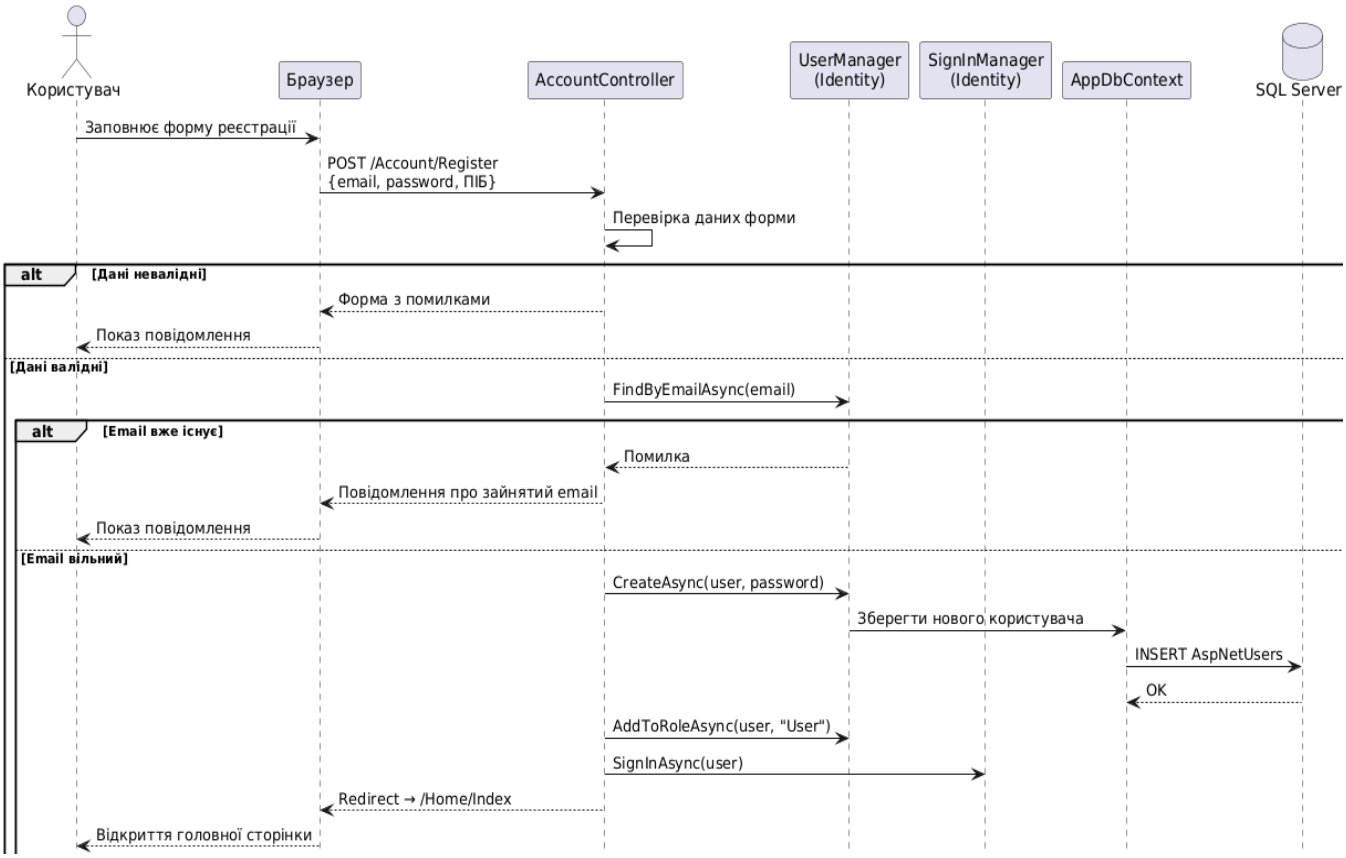

Рисунок 3.3 – Діаграма послідовностей: реєстрація користувача

Додавання товару до кошика виконується через POST-запит на CartController.Add. Користувач натискає кнопку «Додати до кошика», після чого контролер отримує ProductId і кількість товару. Далі викликається CartService, який через AppDbContext перевіряє наявність товару, його залишок на складі та існування CartItem для поточного користувача. Якщо запис уже є, збільшується Quantity. Якщо запису немає, створюється новий CartItem. Після збереження змін користувач перенаправляється назад до сторінки товару або каталогу. Окремо лічильник кошика у шапці сайту може оновлюватися через запит до CartController.Count, який повертає кількість товарів у форматі JSON.

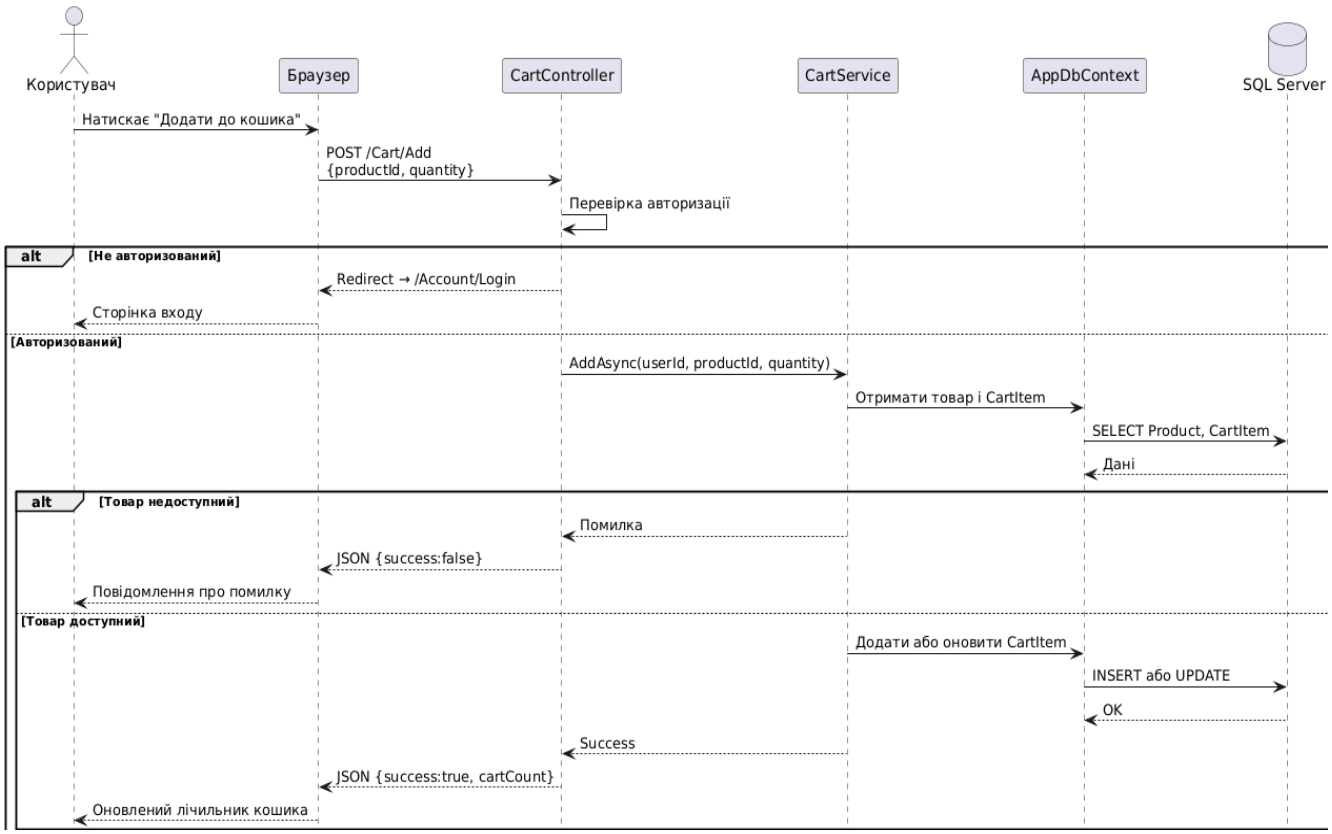

Рисунок 3.4 – Діаграма послідовностей: додавання товару до кошика

Оформлення замовлення починається з форми Checkout. OrderController.Checkout отримує POST-запит із даними доставки та способом оплати. OrderService зчитує поточні CartItems користувача, перевіряє залишки товарів на складі і відкриває транзакцію EF Core. Далі створюється запис Order зі статусом Confirmed, а для кожної позиції кошика формується OrderItem із

фіксованою ціною UnitPrice. Після цього кошик очищається. Якщо на будь-якому етапі виникає помилка, транзакція відкочується і замовлення не створюється частково.

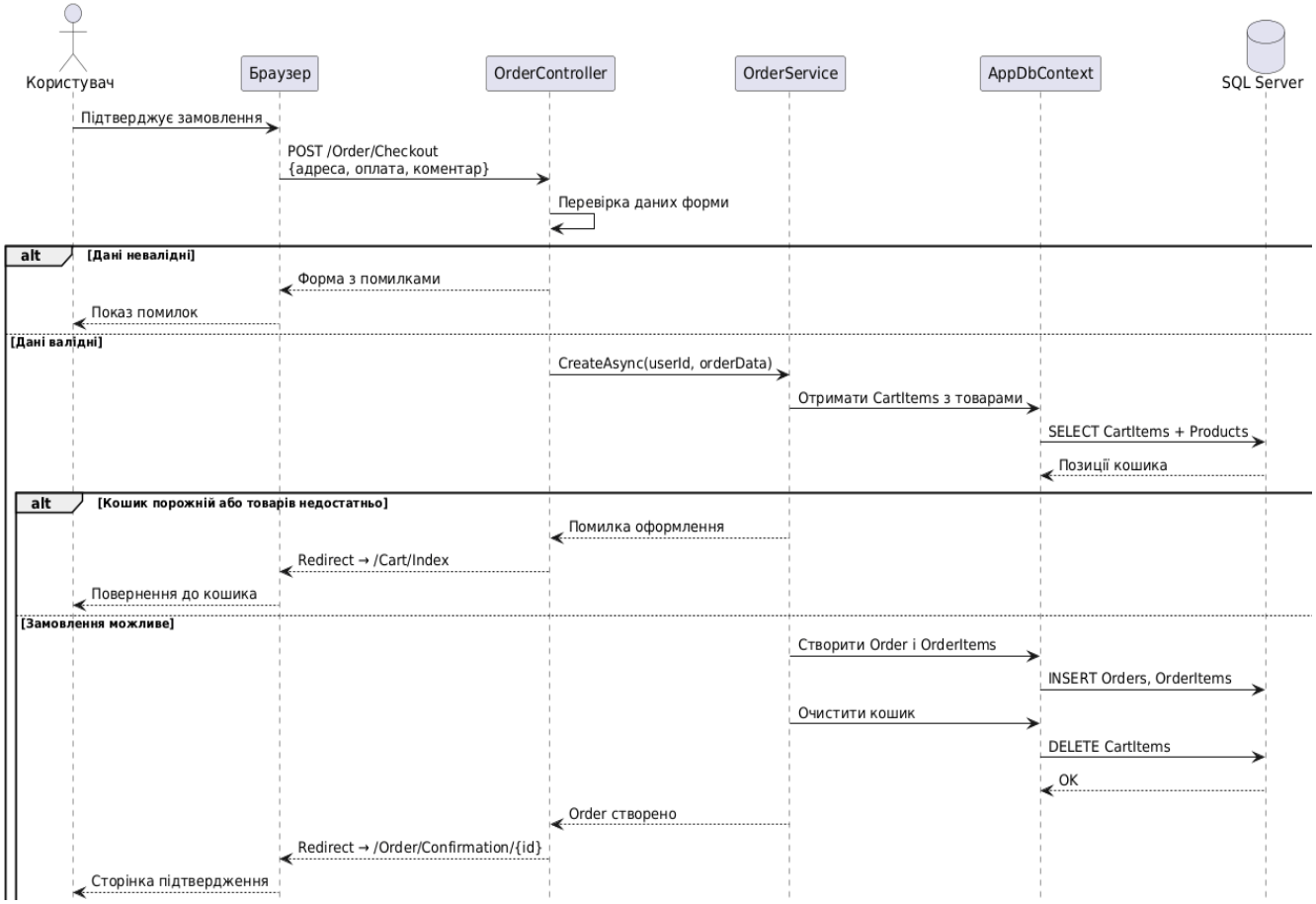

Рисунок 3.5 – Діаграма послідовностей: оформлення замовлення

Діаграми діяльності

Діаграми діяльності описують порядок виконання бізнес-процесів із розгалуженнями та умовами завершення. Розглянуто два сценарії: вхід у систему та пошук товарів у каталозі.

Процес входу починається з форми Login. Користувач вводить email і пароль, після чого дані проходять клієнтську та серверну перевірку. Якщо модель валідна, AccountController передає дані до SignInManager.PasswordSignInAsync. ASP.NET Core Identity порівнює пароль із хешем у базі даних. У разі помилки користувач бачить повідомлення, а кількість невдалих спроб збільшується. Після перевищення

ліміту обліковий запис тимчасово блокується. Якщо перевірка успішна, створюється cookie-сесія і виконується перехід на початкову або запитану сторінку.

Рисунок 3.6 – Діаграма діяльності: процес входу в систему

Пошук і фільтрація виконуються на сторінці каталогу. Користувач задає параметри пошуку, категорії, бренди, сезони, діапазон ціни, сортування і номер

сторінки. ShopController.Index приймає GET-запит і передає дані до CatalogService. Сервіс будує LINQ-запит до Products, додає умови Where для непорожніх параметрів, застосовує сортування і пагінацію через Skip() та Take(). Розмір сторінки становить 12 товарів. Якщо результатів немає, система показує відповідне повідомлення замість порожньої сітки.

Рисунок 3.7 – Діаграма діяльності: пошук та фільтрація товарів

Наведені діаграми уточнюють логіку сценаріїв, які користувач виконує найчастіше. Вони також показують, де саме відбувається перевірка даних, звернення до сервісного шару та повернення результату в інтерфейс.

3.4 Проєктування бази даних

База даних реалізована у Microsoft SQL Server і працює через Entity Framework Core 8.0. Структура охоплює користувачів, каталог товарів, продажі та додаткові клієнтські функції; зміни схеми фіксуються міграціями.

Управління користувачами побудовано на ASP.NET Core Identity. Таблиця AspNetUsers розширена полями FirstName, LastName, Address і RegisteredAt, а ролі зберігаються через AspNetRoles та AspNetUserRoles. Для адрес доставки використовується UserAddresses з полями IsDefault і CreatedAt.

Каталог зосереджений навколо Products. Таблиця містить Name, Description, Price, Stock, ImageUrl, CategoryId, BrandId, SeasonId, HowToUse, Specifications і HasCalculator. Categories, Brands і Seasons виконують роль довідників. ProductBundles описує сумісні товари через MainProductId і RelatedProductId, а Discounts зберігає Percentage, StartDate, EndDate та IsActive.

Продажі представлені таблицями Orders, OrderItems і CartItems. Orders містить TotalAmount, Status, DeliveryAddress, PaymentMethod, IsPaid і Comment. В OrderItems фіксується UnitPrice, тобто ціна товару на момент оформлення, тому подальша зміна ціни в каталозі не впливає на вже створене замовлення. CartItems зберігає UserId, ProductId і Quantity.

Додатковий функціонал винесено в Reviews, WishlistItems, PlannerNotes і StockNotifications. Ці таблиці відповідають за відгуки, список бажань, нотатки садового планувальника та заявки на сповіщення про наявність товару. Цілісність даних підтримується зовнішніми ключами, індексами та правилами видалення.

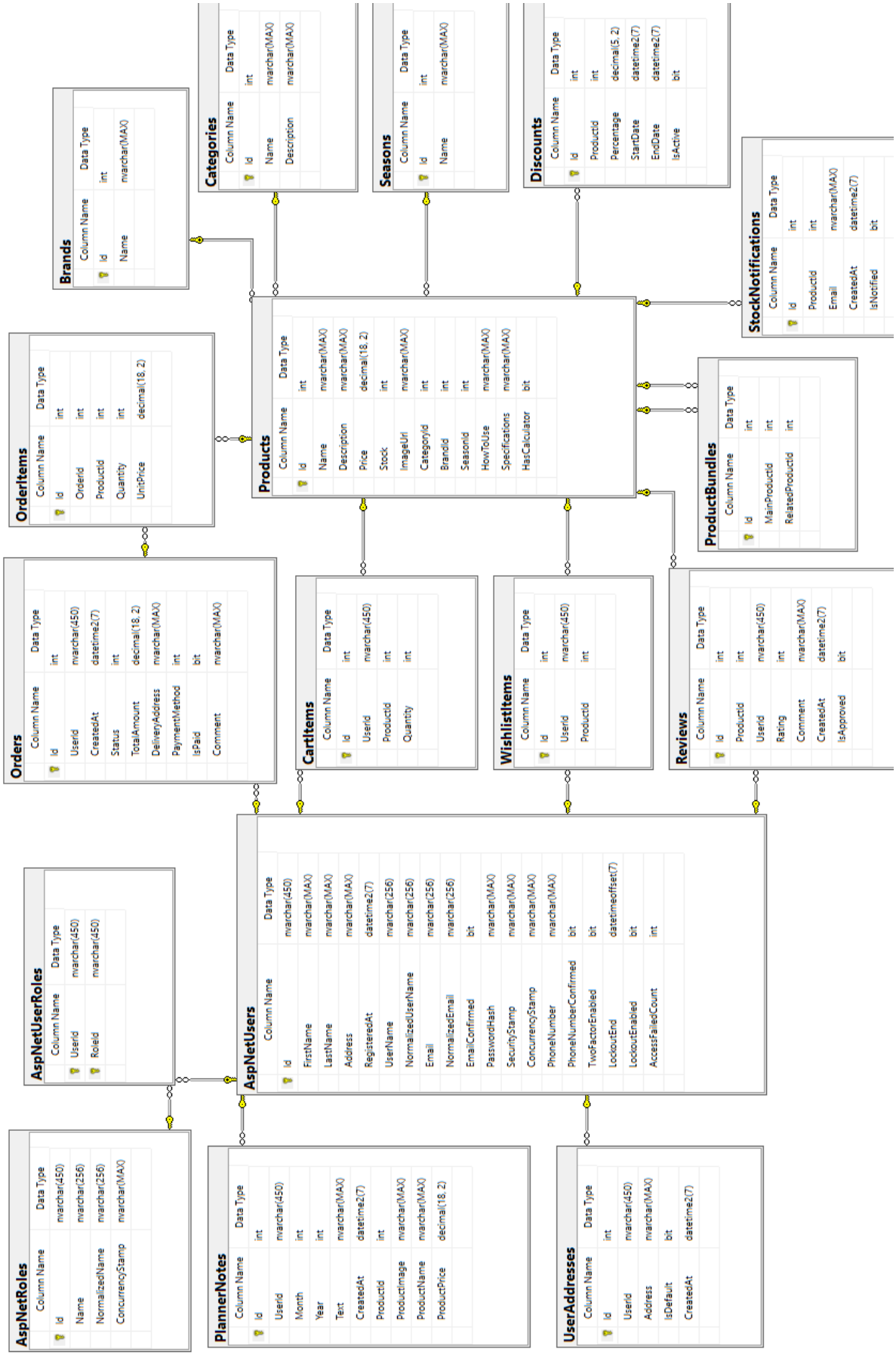

Рисунок 3.8 – ER–діаграма бази даних

ER-діаграма уточнює зв’язки між основними таблицями та показує, як дані користувачів, товарів, кошика і замовлень поєднані між собою. На її основі далі описується програмна реалізація моделей і контексту бази даних.

3.5 Опис інтерфейсів застосунку

Клієнтський інтерфейс GardenTools побудований на Bootstrap 5 і адаптується до різних розмірів екрана, від мобільних пристроїв до десктопних моніторів. Сторінки реалізовані як Razor Views і отримують типізовані дані від контролерів через відповідні ViewModel-класи. Такий підхід дозволяє відокремити логіку підготовки даних від їх відображення у шаблоні.

Головна сторінка (Home)

Головна сторінка реалізована через HomeController і виконує роль початкової точки взаємодії користувача із застосунком. Вона містить кілька функціональних зон: навігаційну панель, банерний блок, секції товарів, тематичні переходи до каталогу та футер.

Навігаційна панель містить логотип Garden Tools, пошуковий рядок і посилання на основні розділи сайту. Через неї користувач може перейти до головної сторінки, каталогу, магазину, планувальника, кошика або особистого кабінету. Кнопка профілю змінює призначення залежно від стану авторизації: авторизований користувач переходить до свого облікового запису, а неавторизований –до форми входу.

Верхній банерний блок реалізовано за допомогою Bootstrap Carousel. Він містить промоційні слайди з короткими текстовими повідомленнями та переходами до відповідних розділів каталогу. Такий блок використовується для швидкої навігації до актуальних товарів і сезонних пропозицій.

Окремі секції головної сторінки відображають товари та категорії, які мають бути помітними для користувача одразу після входу на сайт. Частина товарів підвантажується через ProductsApi, тому дані для секції не дублюються вручну в Razor-шаблоні. Картки товарів містять зображення, назву, ціну, рейтинг і кнопку

переходу до детальної сторінки. Якщо товар має активну знижку, біля акційної ціни може відображатися початкова ціна у перекресленому вигляді.

Секція сезонних пропозицій використовує поточний місяць для визначення актуального сезону. Залежно від цього користувачу пропонуються тематичні товари або перехід до планувальника садових робіт. Сторінку завершує футер із навігаційними посиланнями на інформаційні та службові розділи сайту. Усі блоки головної сторінки оформлені в єдиному стилі GardenTools.

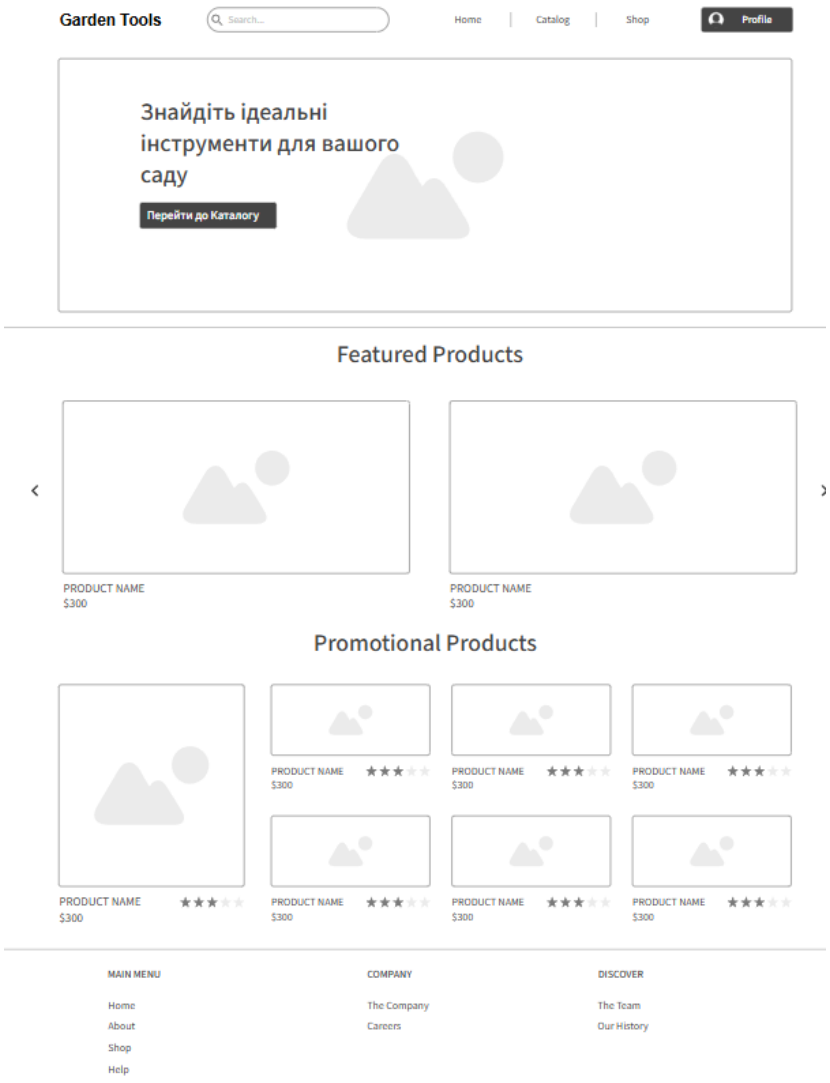

Рисунок 3.9 – Макет головної сторінки GardenTools

Каталог товарів (Shop)

ShopController.Index відображає каталог товарів у вигляді сітки карток і панелі фільтрів. Основна частина сторінки містить перелік товарів, а бічна панель

дає змогу уточнити результати пошуку за кількома параметрами. Користувач може вибрати категорії, бренди, сезони, мінімальну та максимальну ціну, а також спосіб сортування.

Фільтрація реалізована через GET-параметри. Це означає, що вибрані умови зберігаються в URL і можуть бути повторно відкриті через посилання або кнопку «Назад» у браузері. Для кожного непорожнього параметра CatalogService додає відповідну умову до LINQ-запиту. Сортування виконується за назвою, ціною або рейтингом залежно від вибраного значення sortBy.

Кожна картка товару містить зображення, назву, ціну, рейтинг, коротку інформацію про знижку і кнопки для переходу до сторінки деталей або додавання товару до кошика. Каталог використовує пагінацію: на одній сторінці відображається 12 товарів. Навігаційні посилання між сторінками формуються з урахуванням активних фільтрів, щоб користувач не втрачав вибрані умови під час переходу.

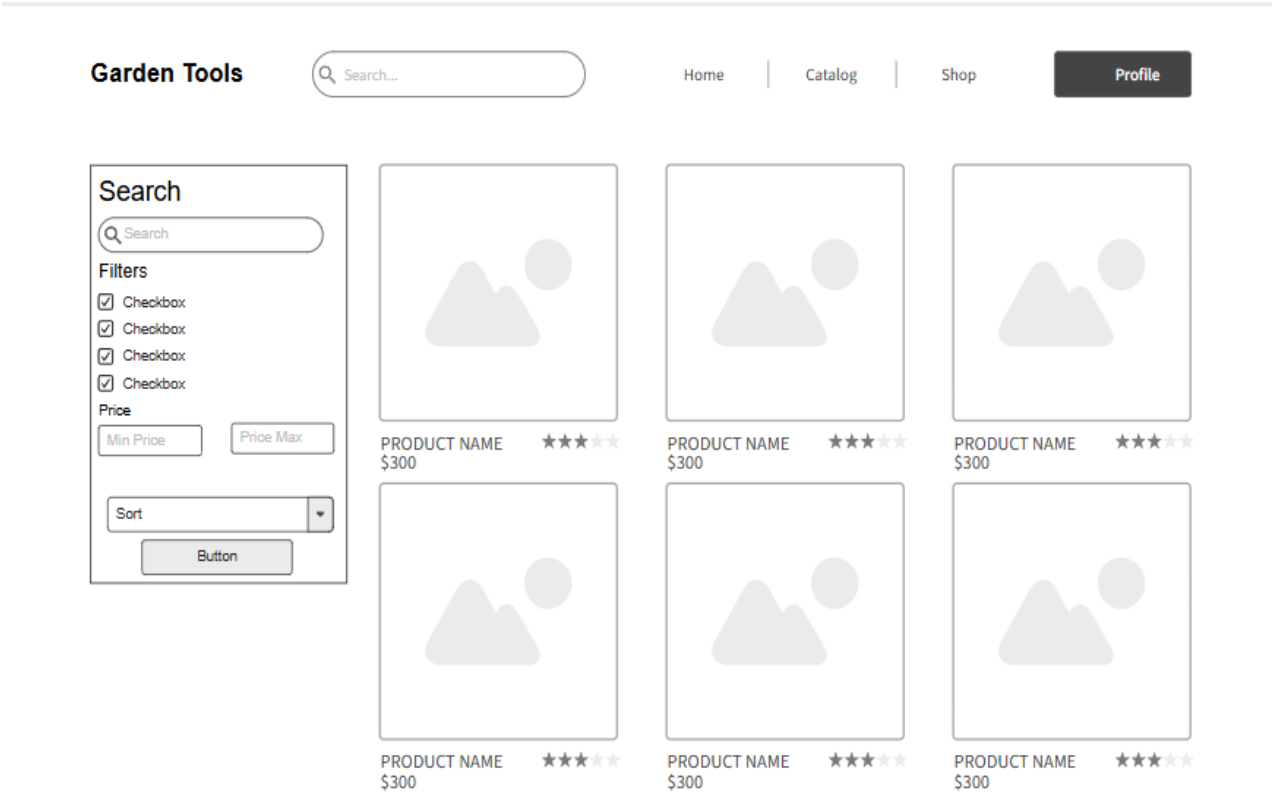

Рисунок 3.10 – Макет сторінки каталогу товарів

Картка товару

Сторінка деталей товару реалізована через ShopController і відкривається за ідентифікатором конкретного товару. Вона відображає назву, зображення, поточну ціну, залишок на складі, опис, технічні характеристики з поля Specifications і інструкцію використання з поля HowToUse. Якщо для товару діє активна знижка, поряд із новою ціною виводиться початкова ціна у перекресленому вигляді.

Окремий блок сторінки присвячений відгукам. У ньому показуються лише записи, для яких IsApproved має значення true. Це дозволяє не виводити на сторінку відгуки, які ще не пройшли перевірку адміністратором. Авторизований користувач може залишити власний відгук, вибравши оцінку і ввівши текстовий коментар.

На сторінці товару також доступне додавання до списку бажань. Для товарів, що мають пов’язані позиції, виводиться блок сумісних товарів із ProductBundles. Якщо товару немає в наявності, користувач може залишити заявку на сповіщення про повернення товару на склад. Така заявка зберігається у StockNotifications.

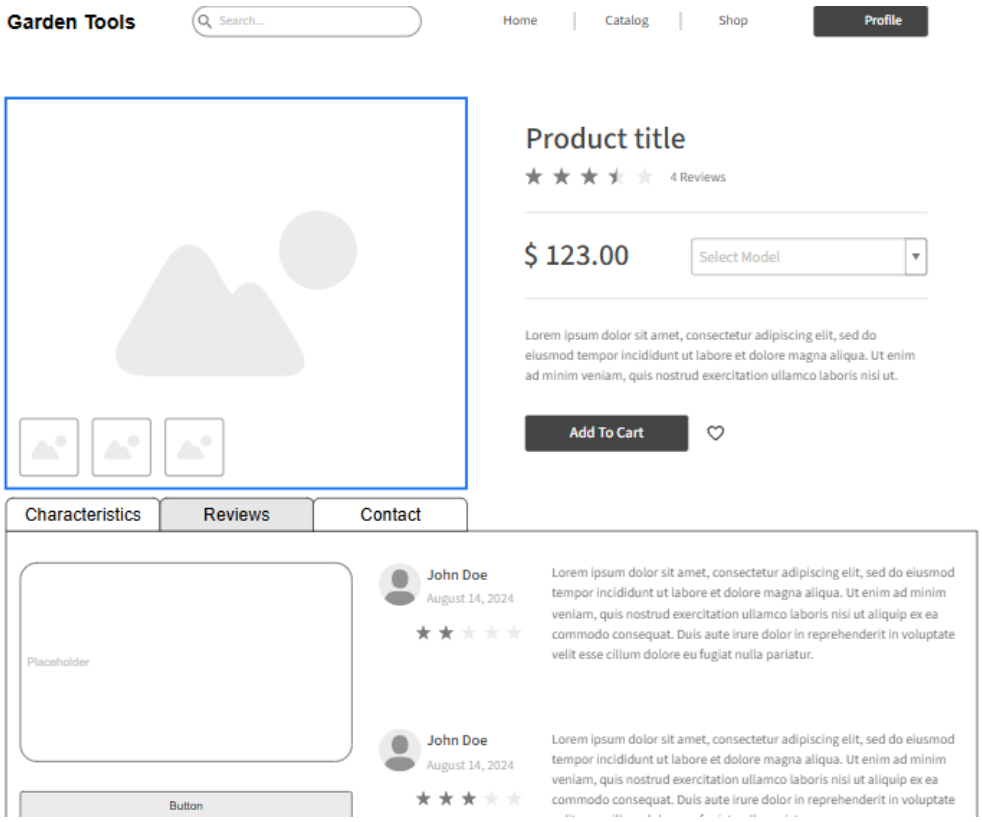

Рисунок 3.11 – Макет картки товару

Особистий кабінет

Особистий кабінет реалізований через AccountController і доступний лише авторизованому користувачу. У ньому зібрано дані профілю, історію замовлень, збережені адреси доставки та швидкі переходи до списку бажань і садового планувальника.

Блок профілю відображає ім’я, прізвище, електронну пошту, адресу і роль користувача. Форма редагування дозволяє змінити персональні дані та оновити пароль. Для зміни пароля користувач має ввести поточний пароль і нове значення, після чого дані проходять перевірку засобами ASP.NET Core Identity.

Окремий блок відповідає за адреси доставки. Користувач може додати нову адресу, видалити зайву або позначити одну з них як основну. Основна адреса автоматично використовується під час оформлення наступного замовлення. У блоці замовлень відображаються останні покупки з датою, сумою, статусом і адресою доставки. Статус показується у вигляді кольорового badge-елемента Bootstrap.

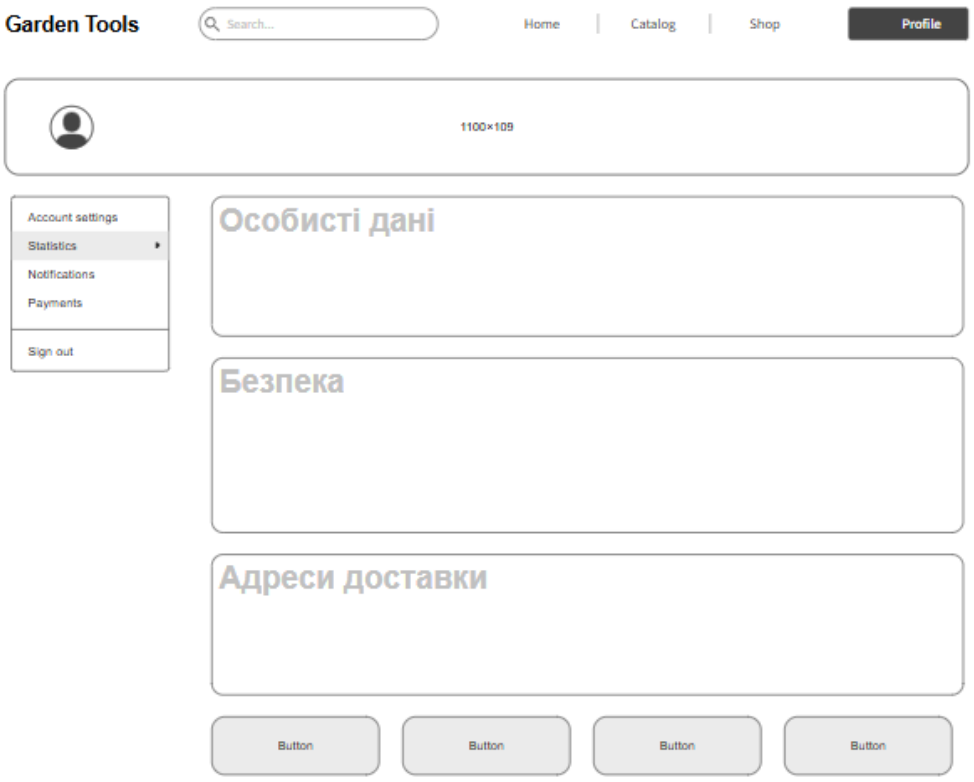

Рисунок 3.12 – Макет сторінки кошика

Кошик та оформлення замовлення

Сторінка кошика формується через CartController.Index і відображає CartItems поточного користувача. Для кожної позиції показано зображення товару, назву, категорію, ціну, залишок на складі, кількість і суму по рядку. У правій частині сторінки розміщено блок підсумку замовлення із загальною кількістю товарів і сумою до сплати.

Зміна кількості товару виконується через POST-запит до CartController.UpdateQuantity зі стандартним оновленням сторінки. Видалення окремої позиції також виконується POST-запитом. Якщо кількість стає некоректною або перевищує залишок на складі, сервіс повертає повідомлення про помилку. Кнопка «Оформити замовлення» доступна лише авторизованому користувачу, оскільки CartController захищений авторизацією.

Форма оформлення замовлення реалізована через OrderController.Checkout. Вона запитує адресу доставки, спосіб оплати і додаткові дані залежно від вибраного методу. Якщо користувач має збережені адреси, поле доставки може бути попередньо заповнене основною адресою. Перед створенням замовлення виконується клієнтська та серверна валідація, після чого OrderService створює замовлення і очищає кошик.

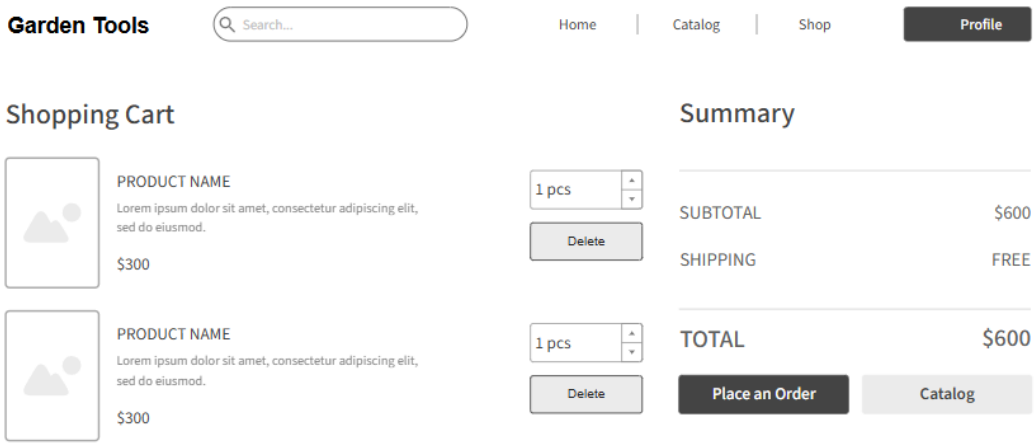

Рисунок 3.13 – Макет особистого кабінету користувача

Усі описані інтерфейси використовують спільну стилістику, навігацію і набір UI-компонентів Bootstrap 5. Завдяки цьому сторінки виглядають узгоджено, а

користувач зберігає однакову логіку взаємодії під час переходу між головною сторінкою, каталогом, карткою товару, кошиком і особистим кабінетом.

Висновки до розділу 3

У третьому розділі описано проєктні рішення для вебзастосунку GardenTools. Архітектура системи подана як трирівнева модель із рівнем представлення, бізнес-логіки та доступу до даних. Razor Views і статичні ресурси відповідають за інтерфейс, сервіси містять бізнес-операції, а AppDbContext та міграції Entity Framework Core працюють із базою даних.

Функціональність застосунку розподілена між контролерами AccountController, AdminController, CartController, HomeController, OrderController, ShopController і WishlistController. Контролери приймають HTTP-запити та передають виконання відповідним сервісам, тому основна бізнес-логіка не зосереджується безпосередньо в них.

Для моделювання системи використано UML-діаграми варіантів використання, класів, послідовностей і діяльності. Вони описують ролі користувачів, доменну модель GardenTools і сценарії роботи системи, зокрема реєстрацію, роботу з кошиком, оформлення замовлення, вхід і пошук товарів.

Базу даних спроєктовано для зберігання користувачів, товарів, кошика, замовлень, відгуків, списку бажань, нотаток планувальника та заявок на сповіщення про наявність товару. Для позицій замовлення передбачено збереження UnitPrice, щоб фіксувати ціну товару на момент купівлі.

Також описано інтерфейси головної сторінки, каталогу, картки товару, особистого кабінету, кошика та форми оформлення замовлення. Отримані архітектурні, модельні та інтерфейсні рішення використано як основу для програмної реалізації, наведеної у розділі 4.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

У четвертому розділі описано практичну реалізацію GardenTools: структуру ASP.NET Core MVC-проєкту, серверні компоненти, клієнтську частину та фрагменти коду, що показують роботу основних функцій. Окрему увагу приділено взаємодії контролерів, сервісів, моделей і бази даних.

4.1 Структура проєкту

GardenTools реалізований як єдиний ASP.NET Core MVC-застосунок із монолітною структурою рішення. Для цього проєкту такий підхід є достатнім, оскільки система має обмежену кількість доменних сутностей і не потребує мікросервісного поділу.

Директорія Controllers відповідає за обробку HTTP-запитів. У проєкті використано сім основних контролерів: AccountController, AdminController, CartController, HomeController, OrderController, ShopController і WishlistController. Кожен із них обслуговує окремий функціональний напрям і передає складніші операції до сервісного шару.

Директорія Data містить інфраструктурні компоненти роботи з базою даних. AppDbContext успадковує IdentityDbContext<ApplicationUser>, містить DbSet-властивості для основних сутностей і налаштовує частину обмежень через Fluent API в OnModelCreating. SeedData використовується для початкового наповнення бази ролями, адміністратором, категоріями, брендами, сезонами та стартовими товарами.

У папці Models розміщено доменні сутності: Product, Category, Brand, Season, Order, OrderItem, CartItem, WishlistItem, Review, Discount, ProductBundle, StockNotification, PlannerNote, UserAddress. Клас ApplicationUser розширює стандартний IdentityUser полями FirstName, LastName, Address і RegisteredAt.

Директорія Services містить бізнес-логіку застосунку. Тут розміщені CartService, OrderService, CatalogService, ProductPricingService, DiscountService, BundleService, ReportService, StockNotificationService і сервіси адміністративної

частини. Через цей шар виконуються перевірки залишків, розрахунок поточної ціни, оформлення замовлення, робота зі знижками, звітами та адміністративними даними.

ViewModels використовуються для передачі підготовлених даних із контролерів у Razor Views. У проєкті є ProductListViewModel, ProductDetailViewModel, CartViewModel, CheckoutViewModel, ProfileViewModel, WishlistViewModel, AdminDashboardViewModel та інші моделі представлення. Це дозволяє не передавати у шаблони зайві поля доменних сутностей.

Директорія Views організована за контролерами. Підпапки Account, Admin, Cart, Home, Order, Shop і Wishlist містять Razor-файли відповідних сторінок. Спільні елементи інтерфейсу розташовані у Views/Shared: основний layout, навігація, футер, адміністративний layout і часткове представлення для валідаційних скриптів.

wwwroot використовується для статичних ресурсів. У ньому розміщені CSS-файли, JavaScript-скрипти, бібліотеки Bootstrap, jQuery, jquery.validate, jquery.validate.unobtrusive та інші клієнтські файли. Конфігураційні файли appsettings.json і appsettings.Development.json містять рядок підключення до SQL Server, а Program.cs реєструє сервіси, AppDbContext, ASP.NET Core Identity і middleware-конвеєр.

Таблиця 4.1 – Ключові файли проєкту GardenTools

Файл / директорія	**Шар**	**Призначення**
AppDbContext.cs	Data	EF Core контекст, DbSet для сутностей, конфігурація Fluent API
SeedData.cs	Data	Початкове наповнення ролей, адміністратора та довідників
Controllers/*.cs	Presentation	Обробка HTTP-запитів і виклик сервісного шару
Services/*.cs	Business Logic	Бізнес-правила, розрахунки, формування даних

Кінець таблиці 4.1

Models/*.cs	Domain	Доменні сутності та ApplicationUser
ViewModels/*.cs	Presentation / DTO	Передача даних між Controller і View
Views//.cshtml	Presentation	Razor Views інтерфейсу
wwwroot/	Presentation	CSS, JavaScript, Bootstrap, jQuery та інші статичні ресурси
Migrations/	Data	EF Core міграції схеми бази даних
Program.cs	Infrastructure	Реєстрація сервісів і налаштування middleware

Показана структура відповідає трирівневій архітектурі, описаній у попередньому розділі. Контролери, сервіси, моделі та представлення розділені за призначенням, тому зміни в одному шарі не потребують повного переписування інших частин проєкту.

4.2 Реалізація серверної частини

Серверна частина GardenTools побудована навколо моделей, контексту бази даних і сервісів. Нижче наведено фрагменти коду, які показують реалізацію каталогу товарів, доступу до бази даних, кошика та оформлення замовлення.

Модель товару Product

Клас Product описує основну сутність каталогу. Він містить назву, опис, ціну, залишок на складі, URL зображення, посилання на категорію, бренд і сезон, а також додаткові поля HowToUse, Specifications і HasCalculator. Зв’язки з Review і Discount дозволяють відображати відгуки та активні знижки для конкретного товару.

Лістинг коду – Доменна модель Product

```
namespace GardenTools.Models
{
    public class Product
    {
        public int Id { get; set; }
```

```
        public string Name { get; set; } = "";
        public string Description { get; set; } = "";
        public decimal Price { get; set; }
        public int Stock { get; set; }
        public string ImageUrl { get; set; } = "";

        public int CategoryId { get; set; }
        public int BrandId { get; set; }
        public int SeasonId { get; set; }

        public Category Category { get; set; } = null!;
        public Brand Brand { get; set; } = null!;
        public Season Season { get; set; } = null!;

        public ICollection<Review> Reviews { get; set; } = new List<Review>();
        public ICollection<Discount> Discounts { get; set; } = new
List<Discount>();

        public string? HowToUse { get; set; }
        public string? Specifications { get; set; }
        public bool HasCalculator { get; set; }
    }
}
```

Конфігурація AppDbContext

AppDbContext успадковує IdentityDbContext<ApplicationUser>, тому до схеми бази даних підключаються таблиці ASP.NET Core Identity. Окрім цього, контекст містить DbSet-властивості для товарів, категорій, замовлень, кошика, відгуків, знижок, списку бажань, планувальника, адрес і сповіщень про наявність. У методі OnModelCreating налаштовано типи decimal-полів і зв’язки ProductBundle.

Лістинг коду – Конфігурація AppDbContext

```
public class AppDbContext : IdentityDbContext<ApplicationUser>
{
    public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }

    public DbSet<Product> Products { get; set; }
    public DbSet<Category> Categories { get; set; }
    public DbSet<Brand> Brands { get; set; }
    public DbSet<Season> Seasons { get; set; }
    public DbSet<Order> Orders { get; set; }
    public DbSet<OrderItem> OrderItems { get; set; }
    public DbSet<CartItem> CartItems { get; set; }
    public DbSet<Review> Reviews { get; set; }
    public DbSet<Discount> Discounts { get; set; }
    public DbSet<WishlistItem> WishlistItems { get; set; }
    public DbSet<StockNotification> StockNotifications { get; set; }
    public DbSet<ProductBundle> ProductBundles { get; set; }
    public DbSet<PlannerNote> PlannerNotes { get; set; }
    public DbSet<UserAddress> UserAddresses { get; set; }
```

```
    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);

        builder.Entity<Product>()
            .Property(p => p.Price)
            .HasColumnType("decimal(18,2)");

        builder.Entity<Discount>()
            .Property(d => d.Percentage)
            .HasColumnType("decimal(5,2)");

        builder.Entity<Order>()
            .Property(o => o.TotalAmount)
            .HasColumnType("decimal(18,2)");

        builder.Entity<OrderItem>()
            .Property(oi => oi.UnitPrice)
            .HasColumnType("decimal(18,2)");

        builder.Entity<ProductBundle>()
            .HasOne(b => b.MainProduct)
            .WithMany()
            .HasForeignKey(b => b.MainProductId)
            .OnDelete(DeleteBehavior.Restrict);

        builder.Entity<ProductBundle>()
            .HasOne(b => b.RelatedProduct)
            .WithMany()
            .HasForeignKey(b => b.RelatedProductId)
            .OnDelete(DeleteBehavior.Restrict);
    }
}
```

CartService: логіка кошика

CartService відповідає за роботу з кошиком авторизованого користувача. Метод AddAsync перевіряє коректність кількості, наявність товару та залишок на складі. Якщо CartItem для поточного користувача і товару вже існує, сервіс збільшує Quantity; якщо ні, створює новий запис..

Лістинг коду – CartService: додавання товару до кошика

```
public async Task<ServiceResult> AddAsync(string userId, int productId, int quantity)
{
    if (quantity <= 0)
        return ServiceResult.Fail("Кількість має бути більшою за 0");

    var product = await _db.Products.FindAsync(productId);

    if (product == null)
        return ServiceResult.Fail("Товар не знайдено");
```

```
    if (product.Stock <= 0)
        return ServiceResult.Fail("Товар відсутній на складі");

    var existing = await _db.CartItems
        .FirstOrDefaultAsync(c => c.UserId == userId && c.ProductId == productId);

    var currentQty = existing?.Quantity ?? 0;
    var newQty = currentQty + quantity;

    if (newQty > product.Stock)
        return ServiceResult.Fail($"На складі лише {product.Stock} шт.");

    if (existing != null)
    {
        existing.Quantity = newQty;
    }
    else
    {
        _db.CartItems.Add(new CartItem
        {
            UserId = userId,
            ProductId = productId,
            Quantity = newQty
        });
    }

    await _db.SaveChangesAsync();
    return ServiceResult.Ok("Товар додано до кошика");
}
```

OrderService: оформлення замовлення

CreateOrderAsync у OrderService створює замовлення на основі поточного кошика. Метод перевіряє наявність позицій, контролює залишки товарів, відкриває транзакцію, резервує склад, створює Order і OrderItems, після чого очищає кошик. Для кожної позиції зберігається UnitPrice, тобто ціна товару на момент оформлення.

Лістинг коду – OrderService: транзакційне оформлення замовлення

```
public async Task<OrderCreationResult> CreateOrderAsync(
    string userId,
    CheckoutViewModel model)
{
    var cartItems = await GetCheckoutCartItemsAsync(userId);

    if (!cartItems.Any())
        return OrderCreationResult.Fail(new[] { "Кошик порожній" });

    var stockErrors = await ValidateStockAsync(cartItems);
    if (stockErrors.Any())
        return OrderCreationResult.Fail(stockErrors);
```

```
        await using var transaction = await _db.Database.BeginTransactionAsync();

        var stockUpdateErrors = await TryReserveStockAsync(cartItems);
        if (stockUpdateErrors.Any())
        {
            await transaction.RollbackAsync();
            return OrderCreationResult.Fail(stockUpdateErrors);
        }

        var order = new Order
        {
            UserId = userId,
            CreatedAt = DateTime.UtcNow,
            Comment = model.Comment,
            Status = OrderStatus.Pending,
            DeliveryAddress = model.DeliveryAddress,
            PaymentMethod = model.PaymentMethod,
            IsPaid = model.PaymentMethod == PaymentMethod.Card,
            TotalAmount = CalculateTotal(cartItems),
            Items = cartItems.Select(i => new OrderItem
            {
                ProductId = i.ProductId,
                Quantity = i.Quantity,
                UnitPrice = _pricingService.GetCurrentPrice(i.Product)
            }).ToList()
        };

        _db.Orders.Add(order);
        _db.CartItems.RemoveRange(cartItems);
        await _db.SaveChangesAsync();
        await transaction.CommitAsync();

        return OrderCreationResult.Ok(order);
    }
```

Наведені фрагменти показують основну серверну логіку GardenTools: опис доменної моделі, конфігурацію доступу до бази даних, додавання товарів до кошика та транзакційне створення замовлення.

4.3 Реалізація клієнтської частини

Клієнтська частина GardenTools побудована на Razor Views, Bootstrap 5 і JavaScript. Razor Views відповідають за серверний рендеринг сторінок, Bootstrap 5 використовується для адаптивної верстки, а JavaScript виконує окремі динамічні дії в інтерфейсі.

У файлі wwwroot/js/site.js реалізовано оновлення лічильника кошика, перемикання видимості пароля та автоматичне закриття повідомлень. Функція

updateCartCount після завантаження сторінки надсилає fetch-запит до CartController.Count, отримує кількість товарів у кошику та оновлює badge-елемент у шапці сайту без перезавантаження сторінки.

Лістинг коду – Оновлення лічильника кошика у site.js

```
async function updateCartCount() {
    try {
        const res = await fetch('/Cart/Count');
        const count = await res.json();
        const badge = document.getElementById('cart-count');
        if (badge) {
            badge.textContent = count;
            badge.style.display = count > 0 ? 'flex' : 'none';
        }
    } catch { }
}

document.addEventListener('DOMContentLoaded', () => {
    updateCartCount();
});
```

Форми реєстрації, входу, редагування профілю та оформлення замовлення перевіряються на сервері через ModelState. Для клієнтської перевірки використовуються Razor Tag Helpers, які генерують HTML-атрибути валідації, а також скрипти jquery.validate і jquery.validate.unobtrusive. Завдяки цьому частина помилок показується біля полів ще до надсилання POST-запиту на сервер.

4.4 Тестування вебзастосунку

Тестування GardenTools виконувалося для перевірки основних функціональних вимог, описаних у розділі 2. Перевірялися сценарії роботи з каталогом, карткою товару, реєстрацією, кошиком, оформленням замовлення, особистим кабінетом та адміністративною панеллю. Частина перевірок виконувалася через інтерфейс застосунку, а бізнес-логіка сервісного шару додатково перевірялася в модулі GardenTools.Tests.

Основна увага приділялася сценаріям, у яких можливі помилки даних: спроба додати до кошика кількість, що перевищує залишок на складі, реєстрація з уже використаною електронною поштою, оформлення замовлення з порожнім

кошиком або некоректними даними доставки. Результати функціональної перевірки наведено у таблиці 4.2.

Таблиця 4.2 – Результати функціонального тестування GardenTools

№	Функціональна вимога	Тестовий сценарій	Очікуваний результат	Статус
1	FR–1.2: Фільтрація каталогу	Вибір категорії, бренду та діапазону ціни	Відображаються тільки товари, що відповідають заданим параметрам	Passed
2	FR–1.3: Пагінація	Каталог із кількістю товарів, що перевищує розмір сторінки	Товари розбиваються на сторінки по 12 позицій	Passed
3	FR–1.4: Порожній результат пошуку	Пошук за рядком, якого немає в назвах або описах товарів	Виводиться повідомлення про відсутність результатів	Passed
4	FR–2.2: Залишок на складі	Перегляд товару зі Stock = 0	Кнопка додавання до кошика недоступна, доступна заявка на сповіщення	Passed
5	FR–3.1: Унікальність email	Реєстрація з уже використаною електронною поштою	Обліковий запис не створюється, форма показує помилку	Passed
6	FR–3.2: Хешування пароля	Перевірка запису користувача після реєстрації	У базі зберігається PasswordHash, відкритий пароль відсутній	Passed
7	FR–4.1: Додавання товару до кошика	Додавання товару авторизованим користувачем	У CartItems створюється або оновлюється запис для користувача і товару	Passed

Кінець таблиці 4.2

8	FR–4.2: Зміна кількості в кошику	Зміна кількості через форму кошика	Кількість і загальна сума перераховуються після POST-запиту	Passed
9	FR–5.2: Оформлення замовлення	Підтвердження замовлення з валідною адресою доставки	Створюється Order зі статусом Pending, кошик очищається	Passed
10	FR–6.1: Особистий кабінет	Перегляд профілю авторизованим користувачем	Відображаються дані профілю, адреси доставки та замовлення	Passed
11	FR–7.1: CRUD товарів	Додавання нового товару через адміністративну панель	Товар зберігається у Products і відображається в каталозі	Passed
12	FR–7.3: Захист адмінпанелі	Перехід до /Admin без ролі Admin	Доступ блокується, користувач перенаправляється до сторінки входу	Passed

Усі наведені перевірки завершилися зі статусом Passed. Окремо перевірено роботу сервісів ProductPricingService, CartService, OrderService, CatalogService, AdminProductService і AdminOrderService. Ці тести підтвердили коректність розрахунку знижок, перевірки залишків, створення замовлення, фільтрації товарів та зміни статусу замовлення адміністратором.

4.5 Управління користувача

Реєстрація та вхід

Новий користувач переходить до форми реєстрації через навігаційну панель. Форма запитує ім’я, прізвище, електронну пошту, пароль, підтвердження пароля та, за потреби, адресу. Для полів використовується клієнтська і серверна валідація:

email має відповідати правильному формату, пароль повинен містити щонайменше 6 символів, а значення пароля і підтвердження мають збігатися.

Після успішної реєстрації ASP.NET Core Identity створює користувача, зберігає пароль у вигляді PasswordHash і призначає роль User. Далі виконується автоматичний вхід, і користувач перенаправляється на головну сторінку. Якщо email уже зареєстрований, система повертає форму з повідомленням про помилку.

Авторизація виконується через форму входу. Користувач вводить email і пароль, після чого SignInManager перевіряє облікові дані. У разі кількох невдалих спроб ASP.NET Core Identity може тимчасово заблокувати обліковий запис. У проєкті встановлено ліміт 5 невдалих спроб і тривалість блокування 5 хвилин.

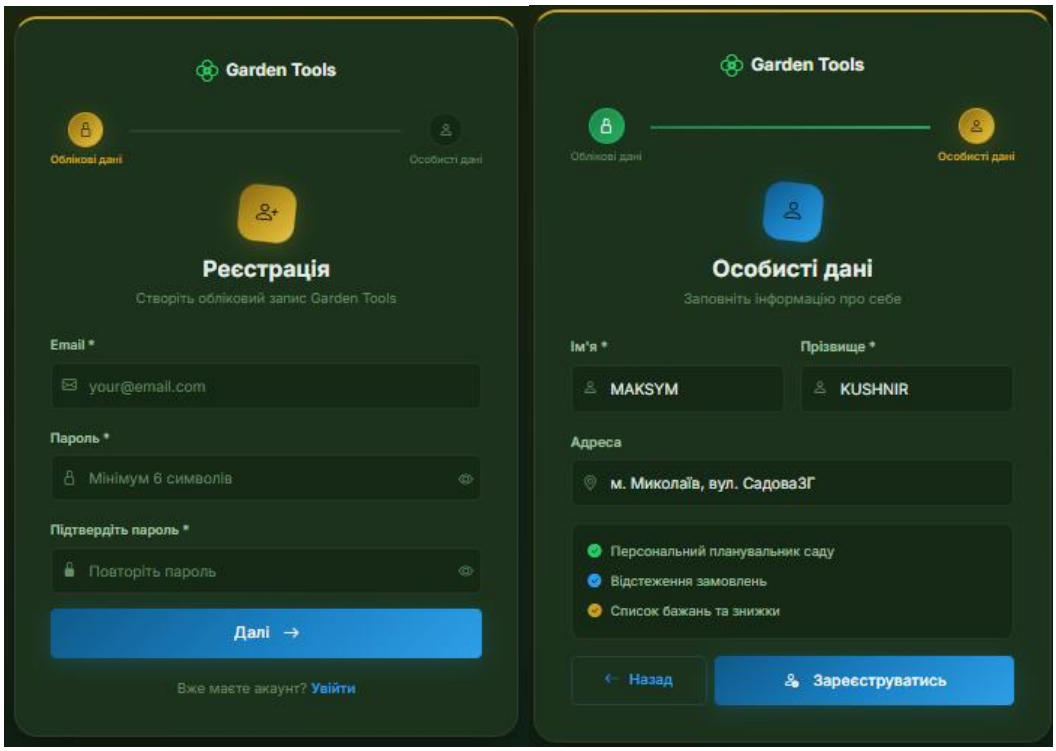

Рисунок 4.1 – Форма реєстрації нового користувача

Перегляд каталогу та пошук товарів

Каталог доступний через розділ Shop. Сторінка складається з панелі фільтрів і сітки товарних карток. Користувач може фільтрувати товари за категорією, брендом, сезоном, мінімальною і максимальною ціною. Додатково доступне сортування за назвою, ціною або рейтингом (рис. 4.2).

Фільтрація виконується через GET-параметри, тому вибрані умови зберігаються в URL. Це дозволяє повернутися до попереднього результату через браузер або передати посилання на відфільтровану сторінку. Пошук за текстом виконується за назвою та описом товару. Якщо результатів немає, сторінка показує повідомлення замість порожньої сітки.

Пагінація розміщена в нижній частині каталогу. На одній сторінці відображається 12 товарів, а навігаційні посилання формуються зі збереженням активних параметрів фільтрації.

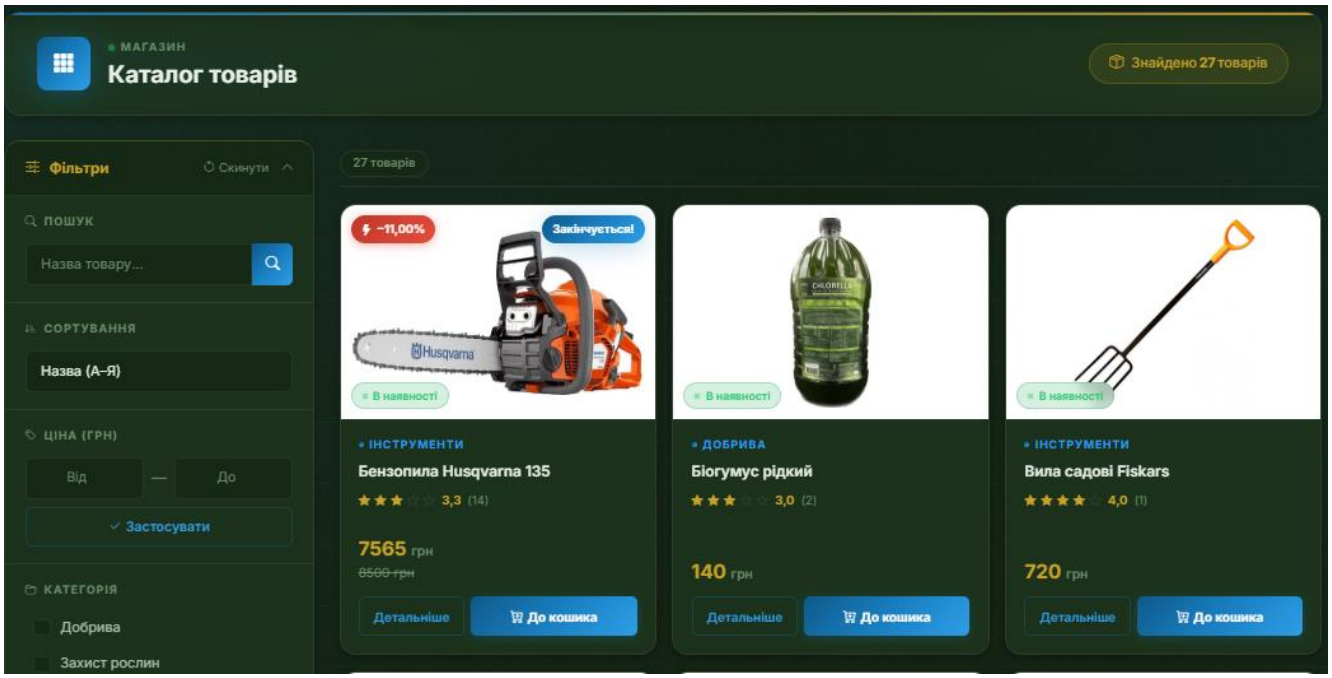

Рисунок 4.2 – Сторінка каталогу з панеллю фільтрів

Картка товару

При переході до деталей товару відкривається сторінка, що містить назву, зображення, поточну ціну, залишок на складі, опис, характеристики з поля Specifications і текст інструкції з поля HowToUse (рис. 4.3). Якщо для товару діє активна знижка, початкова ціна показується у перекресленому вигляді поруч з акційною.

Блок відгуків виводить записи з IsApproved = true. Авторизований користувач може додати власний відгук, вказавши оцінку і коментар. На цій самій сторінці

доступне додавання товару до списку бажань і перегляд сумісних товарів з ProductBundles.

Якщо товар відсутній на складі, замість додавання до кошика користувач може залишити email для сповіщення про повернення товару в наявність. Такий запит зберігається у StockNotifications.

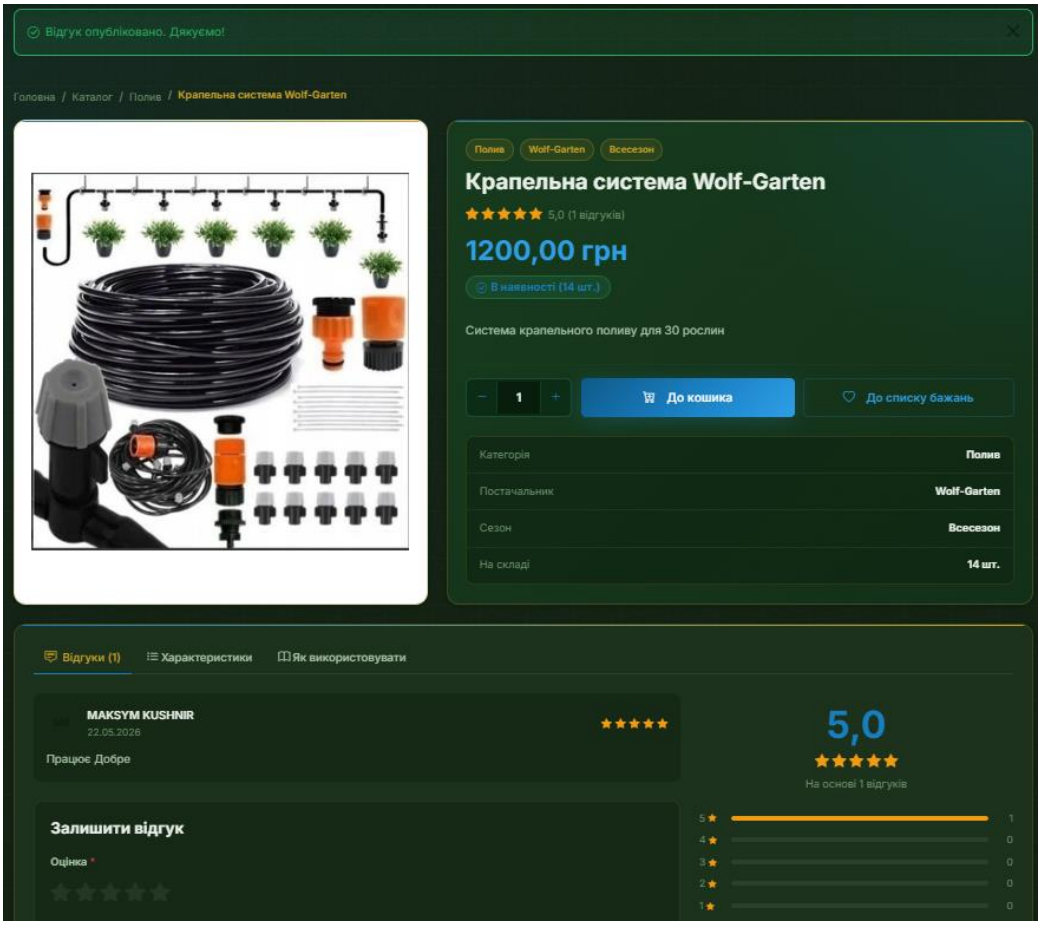

Рисунок 4.3 – Картка товару з відгуками

Управління кошиком

Кошик доступний лише авторизованим користувачам, оскільки CartController захищений атрибутом [Authorize]. Додавання товару виконується через POST-запит до CartController.Add. Якщо товар уже є в кошику, CartService збільшує Quantity; якщо такого запису немає, створюється новий CartItem.

На сторінці кошика відображаються товари поточного користувача: зображення, назва, категорія, ціна, кількість, залишок на складі та сума по рядку (рис. 4.4). Зміна кількості виконується через POST-запит до

CartController.UpdateQuantity зі стандартним оновленням сторінки. Видалення позиції також виконується через POST-запит. Якщо кошик порожній, користувач бачить відповідне повідомлення і посилання на каталог.

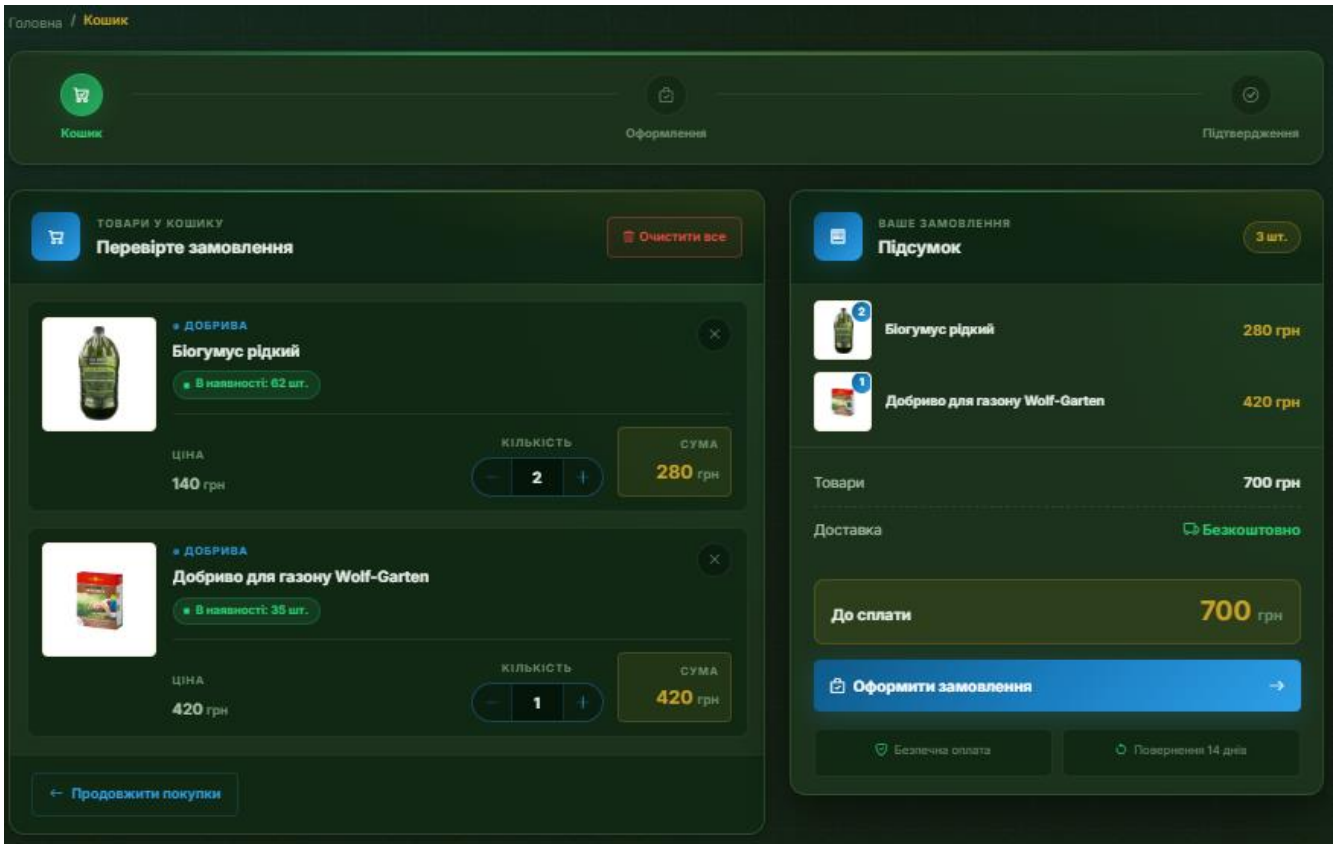

Рисунок 4.4 – Сторінка кошика з підрахунком суми

Оформлення замовлення

Кнопка «Оформити замовлення» переводить користувача до форми Checkout (рис. 4.5). Форма запитує адресу доставки, спосіб оплати і, якщо вибрано оплату карткою, додаткові дані картки. Якщо у користувача є збережені адреси доставки, основна адреса може бути підставлена у форму автоматично.

Після підтвердження OrderService перевіряє кошик і залишки товарів, відкриває транзакцію, резервує склад, створює Order і OrderItems, після чого очищає CartItems поточного користувача. Для кожної позиції зберігається UnitPrice, тобто ціна товару на момент оформлення. Сторінка підтвердження показує номер замовлення, статус, адресу доставки, спосіб оплати і перелік товарів.

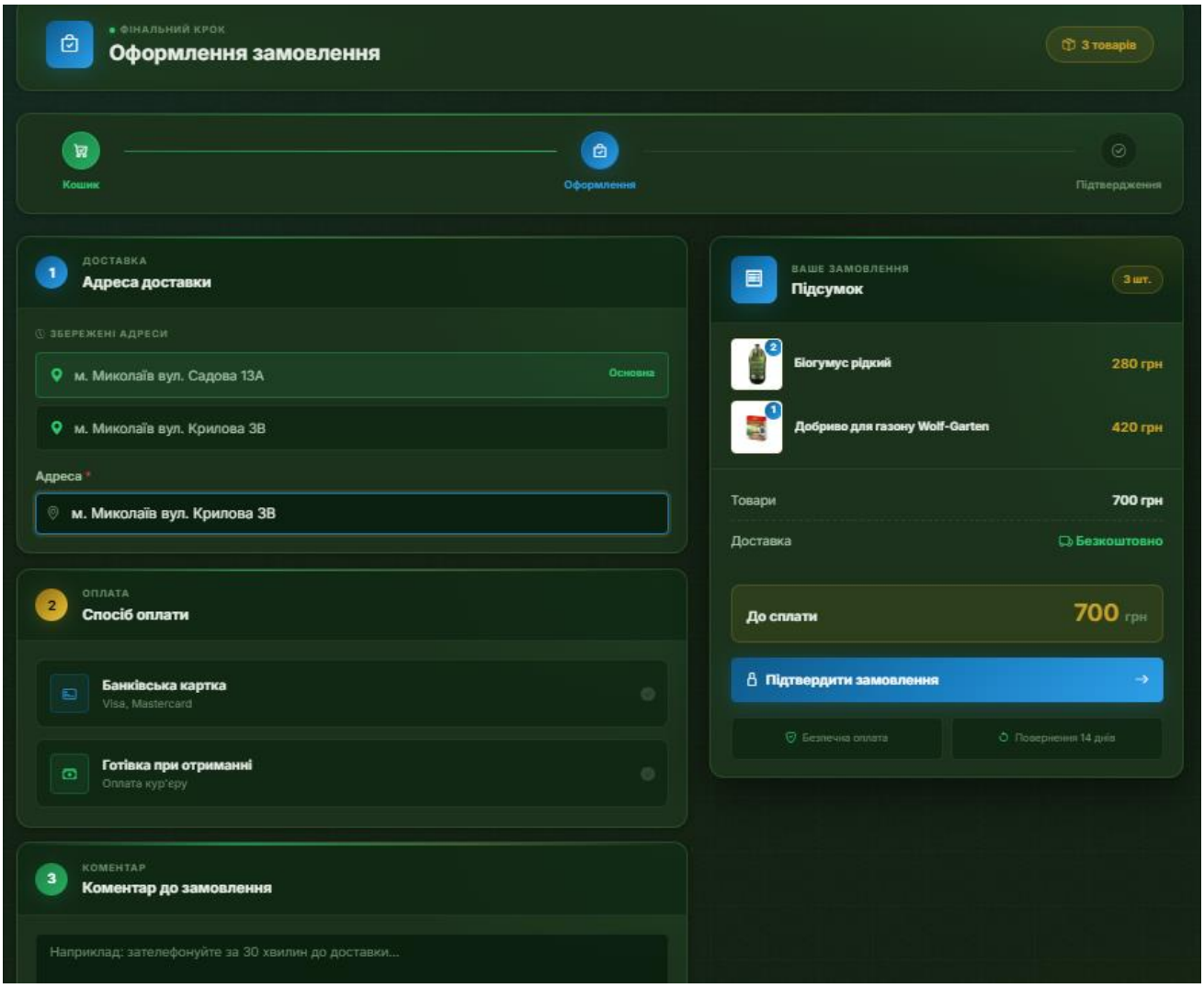

Рисунок 4.5 – Форма оформлення замовлення

Особистий кабінет

Особистий кабінет доступний авторизованому користувачу через меню профілю (рис. 4.6). У ньому відображаються персональні дані, роль, адреса, останні замовлення і швидкі переходи до списку бажань та садового планувальника.

Форма профілю дозволяє змінити ім’я, прізвище, адресу і пароль. Для зміни пароля користувач вводить поточний пароль, новий пароль і підтвердження. Окрема секція відповідає за адреси доставки: користувач може додати нову адресу, видалити існуючу або позначити одну з них як основну.

У блоці замовлень показуються дата, сума, статус і адреса доставки. Статус виводиться у вигляді кольорового badge-елемента. Для перегляду більшої кількості замовлень користувач може перейти до окремої сторінки історії.

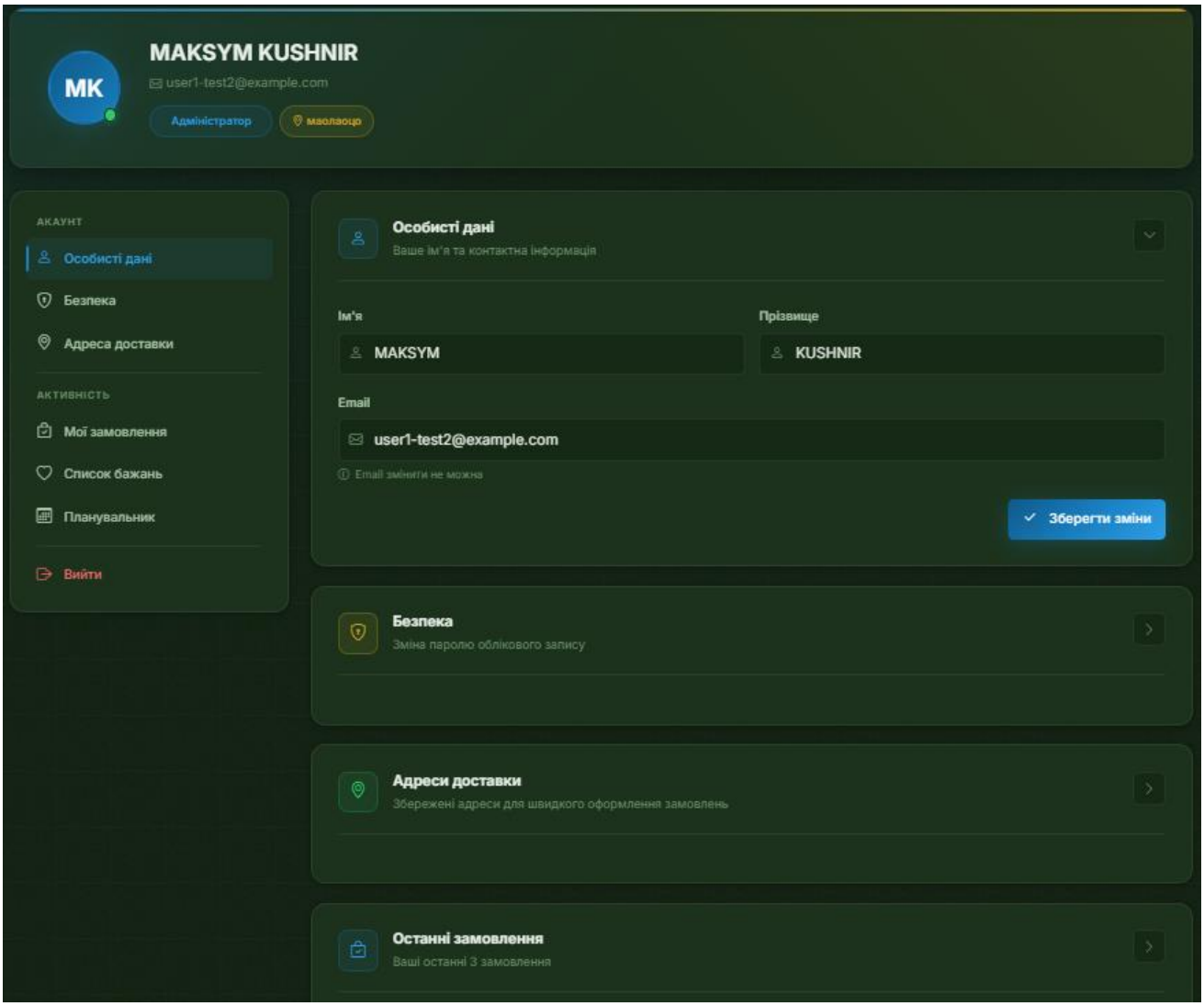

Рисунок 4.6 – Особистий кабінет користувача

Адміністративна панель

Адміністративна панель доступна за адресою /Admin і захищена атрибутом [Authorize(Roles = "Admin")] (рис. 4.7) . Якщо користувач не має відповідної ролі, доступ до панелі блокується.

Дашборд відображає загальну статистику: кількість замовлень, товарів і зареєстрованих користувачів. Розділ управління товарами містить таблицю товарів, пошук і форми створення або редагування. Адміністратор може змінювати назву, опис, ціну, залишок, URL зображення, категорію, бренд, сезон, характеристики та інструкцію використання.

Розділ замовлень показує список замовлень усіх клієнтів. Для кожного замовлення можна переглянути деталі, змінити статус або позначити оплату. У системі використовуються статуси Pending, Confirmed, Shipped, Delivered і Cancelled. Окремі сторінки адміністративної панелі відповідають за категорії, бренди, знижки, сумісні товари, відгуки, користувачів, звіти, аналітику і заявки StockNotifications.

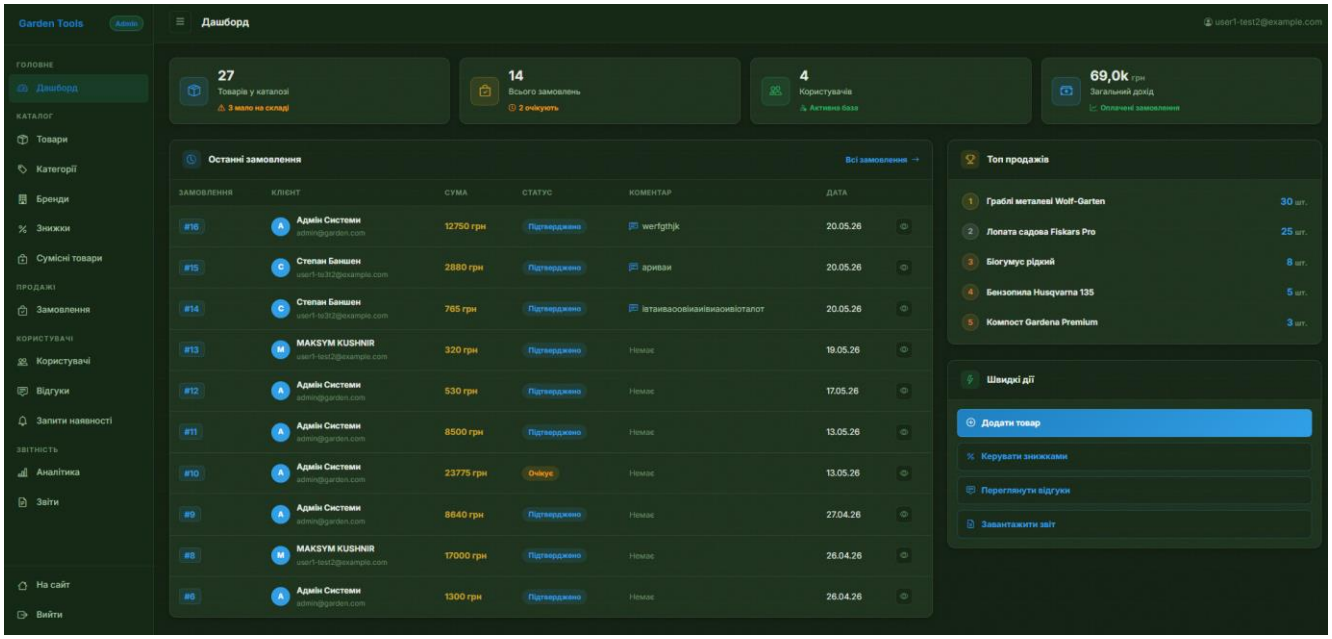

Рисунок 4.7 – Адміністративна панель

Усі основні інтерфейси побудовані в єдиному стилі на Bootstrap 5. Користувач працює з однаковою логікою навігації на сторінках каталогу, товару, кошика, профілю та адміністративної панелі.

Висновки до розділу 4

У четвертому розділі описано програмну реалізацію вебзастосунку GardenTools. Проєкт реалізовано як ASP.NET Core MVC-застосунок із поділом на контролери, сервіси, моделі, ViewModel, Razor Views і статичні ресурси. Така структура відповідає архітектурі, визначеній у попередньому розділі.

Серверна частина побудована навколо AppDbContext, доменних моделей і сервісного шару. Основна бізнес-логіка винесена в CartService, OrderService,

CatalogService, ProductPricingService та адміністративні сервіси. OrderService використовує транзакцію під час оформлення замовлення: перевіряє кошик, резервує залишки товарів, створює Order і OrderItems, після чого очищає кошик користувача.

Окремим рішенням є збереження UnitPrice в OrderItem. Завдяки цьому вартість уже створеного замовлення не змінюється після редагування ціни товару в каталозі. Для інтернет-магазину це має практичне значення, оскільки історія замовлень повинна зберігати фактичну ціну покупки.

Клієнтська частина реалізована на Razor Views, Bootstrap 5 і JavaScript. JavaScript використовується для оновлення лічильника кошика, перемикання видимості пароля та автоматичного закриття повідомлень. Форми реєстрації, входу, профілю й оформлення замовлення перевіряються через серверну валідацію ModelState та клієнтську валідацію jquery.validate і jquery.validate.unobtrusive.

Тестування підтвердило працездатність основних сценаріїв: фільтрації каталогу, перегляду товару, реєстрації, роботи з кошиком, оформлення замовлення, перегляду профілю та доступу до адміністративної панелі. Адміністративна частина дозволяє керувати товарами, категоріями, брендами, знижками, замовленнями, користувачами, відгуками, звітами та заявками StockNotifications. Доступ до неї обмежений роллю Admin через ASP.NET Core Identity.

ВИСНОВКИ

У кваліфікаційній роботі розроблено вебзастосунок GardenTools для продажу інструментів і товарів для садівництва. Система орієнтована на онлайн-перегляд каталогу, пошук товарів, оформлення замовлень і адміністрування асортименту. Результат роботи відповідає поставленій меті: створено спеціалізований інтернет-магазин, який враховує потреби користувачів садівничої тематики.

Під час аналізу предметної області розглянуто особливості онлайн-торгівлі садовими товарами: сезонність попиту, широкий асортимент, різний рівень досвіду покупців і потребу в зручному пошуку. Також виконано порівняння з існуючими платформами Rozetka, Агромаркет, Ідея-сад, Епіцентр та Coira. Порівняння показало, що великі маркетплейси мають розвинену технічну базу, але не орієнтовані саме на садівничу нішу, а спеціалізовані сайти часто поступаються за адаптивністю, зручністю інтерфейсу та функціями особистого кабінету.

На основі проведеного аналізу сформовано функціональні та нефункціональні вимоги до системи. Для реалізації обрано ASP.NET Core MVC на платформі .NET 8.0, мову C#, Entity Framework Core, Microsoft SQL Server, Bootstrap 5, JavaScript ES6+, fetch API та jQuery validation. Такий стек узгоджується з архітектурою проєкту: контролери обробляють HTTP-запити, сервіси містять бізнес-логіку, Razor Views формують інтерфейс, а AppDbContext відповідає за доступ до бази даних.

У роботі спроєктовано архітектуру вебзастосунку, UML-моделі, структуру бази даних і користувацькі інтерфейси. Система побудована за трирівневою моделлю з розділенням рівня представлення, бізнес-логіки та доступу до даних. База даних охоплює користувачів, товари, категорії, бренди, сезони, кошик, замовлення, відгуки, список бажань, нотатки планувальника та заявки на сповіщення про наявність товару. Окремим рішенням є збереження UnitPrice у OrderItems, що фіксує ціну товару на момент оформлення замовлення.

У результаті виконання роботи реалізовано:

– реєстрацію, авторизацію та рольове розмежування доступу;

– каталог товарів із пошуком, фільтрацією, сортуванням і пагінацією;

– сторінку детального перегляду товару з характеристиками, відгуками, знижками та сумісними товарами;

– кошик і оформлення замовлення для авторизованого користувача;

– особистий кабінет із профілем, адресами доставки, історією замовлень, списком бажань і планувальником;

– адміністративну панель для керування товарами, категоріями, брендами, знижками, замовленнями, користувачами, відгуками, звітами та заявками StockNotifications;

– адаптивний інтерфейс на основі Bootstrap 5.

Тестування підтвердило працездатність основних сценаріїв: перегляду каталогу, фільтрації товарів, реєстрації, входу, роботи з кошиком, оформлення замовлення, перегляду профілю та доступу до адміністративної панелі. Перевірка сервісного шару показала коректність розрахунку знижок, контролю залишків, створення замовлення і зміни статусів. Доступ до адміністративних функцій обмежено роллю Admin через ASP.NET Core Identity.

Практичне значення роботи полягає в тому, що GardenTools може бути використаний як основа для інтернет-магазину малого або середнього бізнесу у сфері продажу садових товарів. Застосунок уже містить базові механізми електронної комерції: каталог, кошик, замовлення, профіль користувача та адміністративне керування. Для впровадження у реальну експлуатацію доцільно додати інтеграцію з платіжними сервісами, службами доставки та механізм резервного копіювання бази даних.

Подальший розвиток GardenTools може включати систему рекомендацій, програму лояльності, онлайн-чат із консультантом, push-сповіщення та мобільний застосунок для Android та iOS. Такі розширення не змінюють базову архітектуру системи, але можуть підвищити зручність користування і розширити сценарії взаємодії з покупцями.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Statista. E–commerce worldwide – statistics & facts : website. URL: https://www.statista.com/topics/871/online–shopping (access: 29.04.2026).

2. EVO Group. Ринок e–commerce в Україні : вебсайт. URL: https://evo.business/uk (дата звернення: 29.04.2026).

3. Державна служба статистики України. Сільське господарство України : вебсайт. URL: https://www.ukrstat.gov.ua (дата звернення: 29.04.2026).

4. Nielsen. Consumer behavior in online retail : website. URL: https://www.nielsen.com/insights (access: 29.04.2026).

5. Rozetka. Інтернет-магазин Rozetka : вебсайт. URL: https://rozetka.com.ua (дата звернення: 29.04.2026).

6. Агромаркет. Інтернет-магазин сільськогосподарських товарів : вебсайт. URL: https://agro-market.net/ua (дата звернення: 29.04.2026).

7. Ідея-сад. Інтернет-магазин садових товарів : вебсайт. URL: https://idea-sad.com.ua/ua (дата звернення: 29.04.2026).

8. Епіцентр К. Інтернет-магазин будівельних та садових товарів : вебсайт. URL: https://epicentrk.ua (дата звернення: 29.04.2026).

9. Coira. Інтернет-магазин товарів для саду та городу : вебсайт. URL: https://coira.com.ua/sad-ta-gorod (дата звернення: 29.04.2026).

10. Valiveti S. S. S. Evolution of ASP. NET to ASP. NET Core: Tools, strategies, and implementation approaches. 2025 IEEE 2nd International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS). 2025. p. 1–7.

11. Smith J. P. Entity Framework core in action J. P. Smith. New York : Simon and Schuster, 2021. 528 p.

12. Laaziri M. Analyzing bootsrap and foundation font–end frameworks: a comparative study M. Laaziri, K. Benmoussa, S. Khoulji International Journal of Electrical and Computer Engineering (IJECE). 2019. Vol. 9, № 1.p. 713–722.

13. Kim J. S., Garlan D. Analyzing architectural styles. Journal of Systems and Software. Vol. 83, Issue 7. p. 1216–1235.

14. Gupta P., Mata–Toledo R., Monger M. Utilizing asp. net mvc in web development courses. Journal of Computing Sciences in Colleges. 2012. Vol. 27, Issue 3. p. 10–14.

15. Freeman A., Sanderson S. Pro ASP.NET MVC 5. New York : Apress, 2015. 832 p.

16. Microsoft. ASP.NET MVC Overview : вебсайт. URL: https://learn.microsoft.com/aspnet/mvc (access: 29.04.2026).

17. Microsoft. REST API design : вебсайт. URL: https://learn.microsoft.com/azure/architecture/best–practices/api–design (access: 29.04.2026).

18. Microsoft. Entity Framework Core documentation : вебсайт. URL: https://learn.microsoft.com/ef/core (access: 29.04.2026).

19. Microsoft. SQL Server documentation : вебсайт. URL: https://learn.microsoft.com/sql (access: 29.04.2026).

20. Microsoft. Authentication and authorization in ASP.NET : вебсайт. URL: https://learn.microsoft.com/aspnet/core/security (access: 29.04.2026).

21. Bootstrap Team. Bootstrap Documentation : вебсайт. URL: https://getbootstrap.com/docs (access: 29.04.2026).

22. Freeman A. Pro ASP.NET Core MVC. 2nd ed. New York : Apress, 2018. 1012 p.