Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка програмного забезпечення та тестування вебзастосунку інтернет-магазину з використанням JavaScript–технологій

Виконав(ла): студент(ка) 4 курсу, групи СП-42
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Савіцький А.І. (прізвище та ініціали)

Керівник (підпис) Яворська Є.Б. (прізвище та ініціали)
Нормоконтроль (підпис) Стоянов Ю.М. (прізвище та ініціали)
Завідувач кафедри (підпис) Петрик М.Р. (прізвище та ініціали)
Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра Програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Петрик М.Р.
(підпис) (прізвище та ініціали)
« » 2026 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Савіцькому Андрію Івановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка програмного забезпечення та тестування вебзастосунку інтернет-магазину з використанням JavaScript–технологій

Керівник роботи Яворська Євгенія Богданівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «_06__» *квітня* 2026 року №

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Вимоги замовника, технічні умови, технічне завдання

4. Зміст роботи (перелік питань, які потрібно розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності,			
основи охорони праці			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	**Примітка**
1	Отримання завдання		
2	Аналіз завдання		
3	Виконання розділу “Аналіз вимог до програмної системи”		
4	Виконання розділу “Проєктування та розробка програмної системи”		
5	Виконання розділу “Тестування, впровадження та підтримка”		
6	Виконання розділу “Безпека життєдіяльності, основи охорони праці”		
7	Оформлення пояснювальної записки		
8	Оформлення графічного та презентаційного матеріалу		
9	Попередній захист		
10	Захист		

Студент _____ Савіцький А.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Яворська Є.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка програмного забезпечення та тестування вебзастосунку інтернет-магазину з використанням JavaScript–технологій // Кваліфікаційна робота освітнього рівня «Бакалавр» // Савіцький Андрій Іванович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-42 // Тернопіль, 2026 // С. 77, рис. – 10, табл. – 21, додат. – 3, бібліогр. – 29.

Ключові слова: вебзастосунок; інтернет-магазин; JavaScript; клієнт-серверна архітектура; тестування програмного забезпечення; UML; база даних.

Кваліфікаційна робота присвячена розробці програмного забезпечення та тестуванню вебзастосунку інтернет-магазину з використанням JavaScript-технологій.

У роботі проведено аналіз предметної області електронної комерції та визначено основні вимоги до програмної системи. Виконано моделювання варіантів використання, сформовано функціональні та нефункціональні вимоги, а також здійснено їх пріоритизацію.

У процесі виконання роботи розроблено архітектуру вебзастосунку на основі клієнт-серверної моделі, спроєктовано структуру бази даних та побудовано UML-діаграму класів. Реалізацію програмного забезпечення виконано з використанням сучасних JavaScript-технологій.

Проведено тестування програмного забезпечення, розроблено тестові сценарії та виконано аналіз результатів тестування. Здійснено розгортання програмної системи та визначено системні вимоги до її функціонування. Проведено верифікацію та валідацію вебзастосунку, що підтвердило його відповідність встановленим вимогам.

У результаті виконання роботи створено працездатний вебзастосунок інтернет-магазину, який забезпечує реалізацію основних функцій електронної комерції та може бути використаний у практичній діяльності або як основа для подальшого розвитку.

ABSTRACT

Software Development and Testing of an Online Store Web Application Using JavaScript Technologies // Bachelor's Qualification Work // Andrii Savitskyi // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, Group SP-42 // Ternopil, 2026 // P. 77, fig. – 10, tabl. – 21, app. – 3, bibl. – 29.

Keywords: web application; online store; JavaScript; client-server architecture; software testing; UML; database.

The qualification work is devoted to the development of software and testing of an online store web application using JavaScript technologies.

The paper analyzes the domain of e-commerce and defines the main requirements for the software system. Use case modeling was performed, functional and non-functional requirements were identified and prioritized.

During the development process, the architecture of the web application based on the client-server model was designed, the database structure was developed, and a UML class diagram was created. The software implementation was carried out using modern JavaScript technologies.

Software testing was performed, test cases were developed, and the results of testing were analyzed. The deployment of the system was carried out, and system requirements were defined. Verification and validation of the web application were conducted, confirming its compliance with the specified requirements.

As a result, a functional online store web application was developed, which provides the main features of e-commerce systems and can be used in practical applications or as a basis for further development.

ЗМІСТ

ВСТУП

У сучасних умовах цифрової трансформації економіки та суспільства особливої актуальності набуває розробка ефективних програмних систем для підтримки бізнес-процесів. Одним із ключових напрямів є електронна комерція яка забезпечує здійснення торговельних операцій у мережі Інтернет та дозволяє значно підвищити ефективність взаємодії між продавцями і споживачами. Вебзастосунки інтернет-магазинів є складними програмними системами, що повинні забезпечувати надійність, масштабованість, безпеку та зручність використання.

Розробка сучасного програмного забезпечення повинна здійснюватися з урахуванням міжнародних стандартів у сфері інженерії програмного забезпечення та сучасних науково-технічних підходів. Згідно зі стандартом ISO/IEC/IEEE 12207:2017, процес створення програмного забезпечення включає етапи аналізу вимог, проєктування, реалізації, тестування, впровадження та супроводу. Водночас модель якості програмного забезпечення, визначена стандартом ISO/IEC 25010:2011, передбачає забезпечення таких характеристик, як функціональна придатність, надійність, продуктивність, безпека та зручність використання. У класичних роботах з інженерії програмного забезпечення також підкреслюється важливість системного підходу до розробки програмних продуктів, що включає чітке визначення вимог, моделювання системи та поетапну реалізацію програмного забезпечення.

Сучасні JavaScript-технології (Node.js, React та ін.) забезпечують ефективні засоби для реалізації клієнт-серверної архітектури вебзастосунків, що відповідає принципам модульності, масштабованості та повторного використання компонентів. При цьому важливим етапом життєвого циклу програмного забезпечення є тестування, яке відповідно до стандартів IEEE (зокрема IEEE 829) забезпечує перевірку відповідності програмного продукту встановленим вимогам та виявлення дефектів на різних етапах розробки.

Тому, розробка програмного забезпечення та тестування вебзастосунку інтернет-магазину з використанням JavaScript-технологій є актуальною задачею, що має як теоретичне, так і практичне значення.

Метою кваліфікаційної роботи є розробка програмного забезпечення вебзастосунку інтернет-магазину з використанням сучасних JavaScript-технологій та проведення його тестування відповідно до вимог міжнародних стандартів якості програмного забезпечення.

Для досягнення поставленої мети необхідно вирішити такі задачі:

1. Провести аналіз предметної області електронної комерції з метою визначення основних функцій інтернет-магазину.
2. Проаналізувати існуючі програмні рішення та технології веброзробки з метою обґрунтування вибору JavaScript-технологій.
3. Сформувати функціональні та нефункціональні вимоги до системи з метою визначення її основних характеристик та можливостей.
4. Розробити архітектуру програмного забезпечення з метою забезпечення ефективної роботи вебзастосунку.
5. Спроєктувати структуру бази даних та UML-моделі системи з метою опису логіки роботи та структури даних.
6. Реалізувати програмне забезпечення вебзастосунку з метою створення працездатної системи інтернет-магазину.
7. Розробити інтерфейс користувача з метою забезпечення зручності використання системи.
8. Провести тестування програмного забезпечення з метою перевірки правильності його функціонування.
9. Оцінити результати роботи системи з метою визначення її ефективності.

Об’єктом дослідження є процеси розробки та функціонування вебзастосунків електронної комерції.

Предметом дослідження є методи та засоби розробки і тестування програмного забезпечення вебзастосунку інтернет-магазину з використанням

JavaScript-технологій відповідно до міжнародних стандартів якості програмного забезпечення.

Наукова новизна отриманих результатів полягає у застосуванні стандартів ISO/IEC та IEEE для комплексного підходу до розробки та тестування вебзастосунку інтернет-магазину, а також у побудові узгодженої моделі програмної системи, що враховує вимоги до якості програмного забезпечення відповідно до ISO/IEC 25010 [2] та сучасних підходів до інженерії ПЗ [4], [5].

Практичне значення отриманих результатів полягає у створенні програмного продукту – вебзастосунку інтернет-магазину, який відповідає сучасним вимогам до якості, надійності та безпеки програмного забезпечення. Результати роботи можуть бути використані для впровадження в реальних умовах електронної комерції, а також у навчальному процесі для підготовки фахівців у галузі інженерії програмного забезпечення.

Апробація результатів кваліфікаційної роботи здійснювалася шляхом тестування розробленого програмного забезпечення, а також демонстрації його функціональних можливостей у навчальному процесі.

Публікації за темою кваліфікаційної роботи відсутні.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

У процесі розробки програмного забезпечення важливим етапом є аналіз вимог до майбутньої системи. Саме на цьому етапі визначаються основні функціональні та нефункціональні вимоги, досліджується предметна область, виявляються потреби користувачів і формуються критерії ефективності програмного продукту.

1.1 Аналіз предметної області електронної комерції та вебзастосунків інтернет-магазинів

Електронна комерція є одним із найбільш динамічно розвинених напрямів сучасної цифрової економіки, що забезпечує здійснення торговельних операцій із використанням інформаційно-комунікаційних технологій. Основною складовою електронної комерції є інтернет-магазини, які представляють собою вебзастосунки, призначені для автоматизації процесів продажу товарів і послуг через мережу Інтернет.

Сучасні інтернет-магазини забезпечують реалізацію основних бізнес-процесів, таких як представлення товарів у вигляді каталогу, пошук і фільтрація продукції, формування кошика покупця, оформлення замовлення, а також обробка даних користувачів. Ефективність функціонування таких систем значною мірою залежить від правильності їх проєктування та відповідності вимогам користувачів [4].

Вебзастосунки інтернет-магазинів, як правило, реалізуються на основі клієнт-серверної архітектури, що передбачає розподіл функцій між клієнтською частиною (інтерфейс користувача) та серверною частиною (обробка даних, бізнес-логіка, взаємодія з базою даних). Такий підхід забезпечує масштабованість, гнучкість та можливість подальшого розвитку програмної системи [5].

Таким чином, розробка програмного забезпечення вебзастосунку інтернет-магазину потребує комплексного підходу, що включає аналіз предметної області, визначення вимог до системи та вибір ефективних технологічних рішень.

1.2 Постановка завдання та цілей розробки програмного забезпечення

Основною задачею даної кваліфікаційної роботи є розробка програмного забезпечення вебзастосунку інтернет-магазину, який забезпечує автоматизацію процесів електронної комерції та відповідає сучасним вимогам до якості програмного забезпечення.

Відповідно до стандарту ISO/IEC/IEEE 12207:2017, процес розробки програмного забезпечення включає етапи аналізу вимог, проєктування, реалізації, тестування та впровадження [1]. Крім того, відповідно до стандарту ISO/IEC 25010:2011, програмна система повинна відповідати вимогам функціональної придатності, надійності, продуктивності та безпеки [2].

Метою розробки є створення вебзастосунку, який забезпечує:

- перегляд каталогу товарів;
- реєстрацію та авторизацію користувачів;
- формування кошика;
- оформлення замовлення;
- адміністрування системи.

Реалізація зазначених функцій дозволить забезпечити ефективне функціонування інтернет-магазину та задовольнити потреби користувачів.

1.3 Пошук акторів та варіантів використання системи

Для формалізації функціональних вимог до програмної системи використовується підхід моделювання варіантів використання (Use Case), який дозволяє описати взаємодію користувачів із системою [4].

Основними акторами системи є:

– гість (неавторизований користувач);

– зареєстрований користувач;

– адміністратор.

Для більш наочного представлення взаємодії користувачів із системою сформовано таблицю акторів та відповідних варіантів використання (табл. 1.1).

Таблиця 1.1 – Актори та варіанти використання вебзастосунку інтернет-магазину

Актор	**Варіант використання**	**Опис**
Гість (неавторизований користувач)	Перегляд каталогу товарів	Перегляд списку доступних товарів у системі
	Пошук товарів	Виконання пошуку за назвою або категорією
	Перегляд інформації про товар	Отримання детальної інформації про товар
	Реєстрація	Створення нового облікового запису
Користувач (зареєстрований)	Авторизація	Вхід у систему за логіном і паролем
	Перегляд каталогу товарів	Доступ до повного каталогу продукції
	Додавання товару до кошика	Формування списку покупок
	Редагування кошика	Зміна кількості або видалення товарів
	Оформлення замовлення	Завершення процесу покупки
	Перегляд історії замовлень	Доступ до інформації про попередні покупки
Адміністратор	Авторизація	Вхід у систему з правами адміністратора
	Управління товарами	Додавання, редагування, видалення товарів
	Управління користувачами	Перегляд і редагування даних користувачів
	Управління замовленнями	Обробка та контроль замовлень
	Формування звітів	Отримання статистичної інформації

Наведена таблиця дозволяє систематизувати функціональні можливості програмної системи та є основою для побудови UML-діаграми варіантів використання [22].

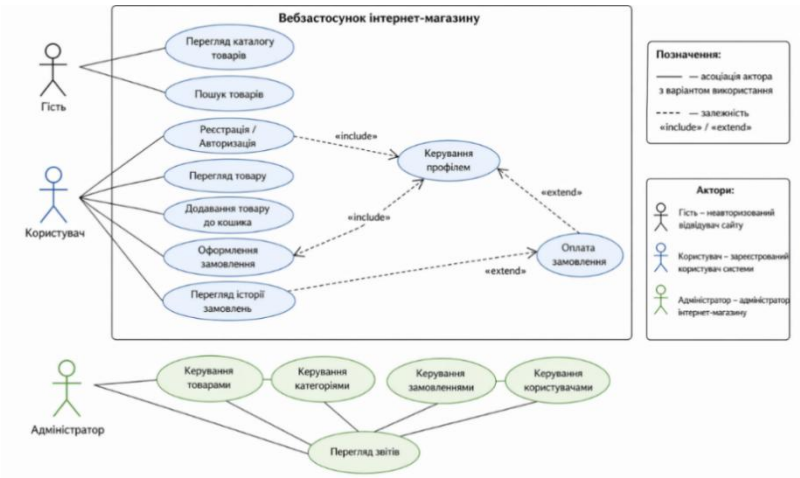

Рисунок 1.1 – UML-діаграма варіантів використання вебзастосунку інтернет-магазину

На рисунку 1.1 представлено UML-діаграму варіантів використання вебзастосунку інтернет-магазину, яка відображає взаємодію акторів із системою.

Основними акторами системи є гість, зареєстрований користувач та адміністратор. Гість має обмежений доступ до функціональності системи та може здійснювати перегляд каталогу товарів, пошук продукції та ознайомлення з інформацією про окремі товари. Для отримання доступу до розширених можливостей системи гість повинен пройти процедуру реєстрації.

Зареєстрований користувач після авторизації отримує доступ до повного функціоналу системи, зокрема можливості додавання товарів до кошика, редагування його вмісту, оформлення замовлення, а також перегляду історії своїх покупок. Цим забезпечується реалізація основних бізнес-процесів електронної комерції.

Адміністратор системи виконує функції управління та супроводу вебзастосунку. До його основних можливостей належать управління товарами, користувачами та замовленнями, а також формування звітної інформації. Це забезпечує підтримку актуальності даних та ефективне функціонування програмної системи.

Зв’язки між акторами та варіантами використання відображають доступні сценарії взаємодії із системою. Деякі варіанти використання можуть бути

пов’язані між собою відношеннями включення або розширення, що дозволяє деталізувати поведінку системи в окремих сценаріях.

Побудована UML-діаграма варіантів використання дозволяє формалізувати функціональні вимоги до програмної системи, визначити основні сценарії її використання та слугує основою для подальшого проєктування архітектури та реалізації вебзастосунку.

1.4 Опис ключових варіантів використання вебзастосунку інтернет-магазину

Основними варіантами використання системи є:

- реєстрація користувача;
- авторизація користувача;
- перегляд каталогу товарів;
- пошук товарів;
- додавання товару до кошика;
- оформлення замовлення;
- управління товарами.

Опис цих варіантів дозволяє деталізувати функціональні вимоги та забезпечити їх подальшу реалізацію.

1.5 Формування функціональних та нефункціональних вимог

На основі проведеного аналізу сформовано функціональні та нефункціональні вимоги до програмної системи (табл. 1.2 та 1.3).

Таблиця 1.2 – Функціональні вимоги до вебзастосунку

№	Функціональна вимога	Опис
F1	Реєстрація користувача	Система повинна забезпечувати можливість створення облікового запису
F2	Авторизація користувача	Система повинна забезпечувати вхід користувача за

		логіном і паролем
F3	Перегляд каталогу товарів	Система повинна відображати список доступних товарів
F4	Пошук товарів	Система повинна забезпечувати пошук товарів за ключовими словами
F5	Перегляд інформації про товар	Система повинна надавати детальну інформацію про товар
F6	Додавання товару до кошика	Користувач повинен мати можливість додавати товари до кошика
F7	Редагування кошика	Система повинна дозволяти зміну кількості або видалення товарів
F8	Оформлення замовлення	Система повинна забезпечувати оформлення замовлення
F9	Перегляд історії замовлень	Користувач повинен мати доступ до своїх попередніх замовлень
F10	Управління товарами	Адміністратор повинен мати можливість додавати, редагувати та видаляти товари
F11	Управління користувачами	Адміністратор повинен мати можливість переглядати та редагувати користувачів
F12	Управління замовленнями	Адміністратор повинен мати можливість обробляти замовлення

Таблиця 1.3 – Нефункціональні вимоги до програмної системи

№	**Нефункціональна вимога**	**Опис**
NF1	Продуктивність	Система повинна забезпечувати швидке завантаження сторінок (до 2–3 секунд)
NF2	Масштабованість	Система повинна підтримувати збільшення кількості користувачів
NF3	Надійність	Система повинна стабільно працювати без збоїв
NF4	Безпека	Система повинна забезпечувати захист даних користувачів
NF5	Зручність використання	Інтерфейс повинен бути інтуїтивно зрозумілим
NF6	Сумісність	Система повинна працювати у сучасних браузерах
NF7	Підтримуваність	Програмний код повинен бути структурованим та зрозумілим
NF8	Доступність	Система повинна бути доступною 24/7
NF9	Розширюваність	Система повинна підтримувати додавання нових функцій
NF10	Відмовостійкість	Система повинна мінімізувати втрату даних при помилках

Сформульовані вимоги визначають основні характеристики програмної системи та відповідають моделі якості програмного забезпечення ISO/IEC 25010 [2, 11].

1.6 Простежуваність та пріоритизація вимог

Для забезпечення узгодженості між сформульованими вимогами, варіантами використання та програмною реалізацією доцільно побудувати таблицю простежуваності вимог (див. табл. 1.4). Така таблиця дозволяє

встановити відповідність між функціональними та нефункціональними вимогами, сценаріями взаємодії користувачів із системою та основними програмними модулями.

Таблиця 1.4 – Простежуваність вимог: «Вимога → Use Case → Модуль системи»

Код вимоги	**Вимога**	**Відповідний Use Case**	**Модуль системи**
1	**2**	**3**	**4**
F1	Реєстрація користувача	Реєстрація користувача	Модуль автентифікації та реєстрації
F2	Авторизація користувача	Авторизація користувача	Модуль автентифікації та реєстрації

Продовження таблиці 1.4

1	**2**	**3**	**4**
F3	Перегляд каталогу товарів	Перегляд каталогу товарів	Модуль каталогу товарів
F4	Пошук товарів	Пошук і фільтрація товарів	Модуль пошуку та фільтрації
F5	Перегляд інформації про товар	Перегляд інформації про товар	Модуль каталогу товарів
F6	Додавання товару до кошика	Додавання товару до кошика	Модуль кошика
F7	Редагування кошика	Редагування кошика	Модуль кошика
F8	Оформлення замовлення	Оформлення замовлення	Модуль замовлень
F9	Перегляд історії замовлень	Перегляд історії замовлень	Модуль облікового запису користувача
F10	Управління товарами	Управління товарами	Адміністративний модуль
F11	Управління користувачами	Управління користувачами	Адміністративний модуль
F12	Управління замовленнями	Управління замовленнями	Адміністративний модуль / модуль замовлень
NF1	Продуктивність	Усі варіанти використання	Усі модулі системи
NF4	Безпека	Реєстрація, авторизація, оформлення замовлення	Модуль автентифікації, модуль замовлень
NF5	Зручність використання	Перегляд каталогу, пошук, кошик, оформлення замовлення	Інтерфейс користувача
NF7	Підтримуваність	Усі варіанти використання	Архітектура та програмні модулі системи
NF9	Розширюваність	Усі варіанти використання	Архітектура системи

Наведена таблиця простежуваності вимог дозволяє встановити зв’язок між етапами аналізу, проєктування та реалізації програмної системи. Це забезпечує цілісність розробки, полегшує контроль повноти реалізації вимог і створює основу для подальшого тестування вебзастосунку.

Для визначення важливості вимог до програмної системи доцільно застосувати метод MoSCoW [12], який передбачає їх поділ на чотири категорії: обов’язкові (Must have), бажані (Should have), можливі (Could have) та необов’язкові (Won’t have). Такий підхід дозволяє встановити пріоритети реалізації функціональності системи та оптимізувати процес розробки.

Таблиця 1.5 – Пріоритизація вимог за методом MoSCoW

Код вимоги	**Вимога**	**Пріоритет**	**Опис**
F1	Реєстрація користувача	Must have	Базова функція для роботи системи
F2	Авторизація користувача	Must have	Забезпечує доступ до персоналізованих функцій
F3	Перегляд каталогу товарів	Must have	Основна функція інтернет-магазину
F4	Пошук товарів	Must have	Забезпечує зручність користування
F5	Перегляд інформації про товар	Must have	Необхідна для прийняття рішення про покупку
F6	Додавання товару до кошика	Must have	Ключова функція покупки
F7	Редагування кошика	Must have	Забезпечує гнучкість оформлення замовлення
F8	Оформлення замовлення	Must have	Завершальний етап покупки
F9	Перегляд історії замовлень	Should have	Додаткова функціональність для користувача
F10	Управління товарами	Must have	Необхідна функція адміністратора
F11	Управління користувачами	Should have	Допоміжна функція адміністрування
F12	Управління замовленнями	Must have	Ключова функція адміністрування
NF1	Продуктивність	Must have	Забезпечує швидку роботу системи
NF4	Безпека	Must have	Захист персональних даних користувачів
NF5	Зручність використання	Must have	Забезпечує комфортну взаємодію
NF2	Масштабованість	Should have	Важлива при збільшенні навантаження
NF9	Розширюваність	Could have	Дозволяє додавати нові функції
NF10	Відмовостійкість	Should have	Забезпечує стабільність системи

Проведена пріоритизація вимог дозволяє визначити ключову функціональність вебзастосунку інтернет-магазину, яка повинна бути реалізована в першу чергу. Вимоги категорії Must have є критично важливими для роботи системи, тоді як Should have та Could have можуть бути реалізовані на наступних етапах розвитку програмного забезпечення [4, 5].

Висновки до розділу 1

У першому розділі кваліфікаційної роботи проведено аналіз предметної області електронної комерції та особливостей функціонування вебзастосунків інтернет-магазинів. Встановлено, що сучасні системи електронної комерції повинні забезпечувати ефективну реалізацію основних бізнес-процесів, зокрема управління товарами, обробку замовлень та взаємодію з користувачами.

Виконано постановку задачі розробки програмного забезпечення вебзастосунку інтернет-магазину та визначено основні цілі його створення відповідно до сучасних вимог інженерії програмного забезпечення та міжнародних стандартів [1], [2].

На основі аналізу визначено основних акторів системи та сформовано варіанти її використання, що дозволило формалізувати функціональні вимоги до програмної системи. Побудована UML-діаграма варіантів використання відображає основні сценарії взаємодії користувачів із системою та є основою для подальшого проєктування.

Сформовано функціональні та нефункціональні вимоги до вебзастосунку, які визначають основні характеристики програмного продукту відповідно до моделі якості ISO/IEC 25010 [2]. Також побудовано таблицю простежуваності вимог, що забезпечує узгодженість між етапами аналізу, проєктування та реалізації системи.

Крім того, проведено пріоритизацію вимог за методом MoSCoW, що дозволило визначити критично важливі функції системи та оптимізувати процес її розробки.

У розділі сформовано повний набір вимог до програмної системи, що є основою для подальшого проєктування та реалізації вебзастосунку інтернет-магазину.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

Після проведення аналізу предметної області та визначення вимог до програмної системи наступним етапом є її проєктування та розробка. На цьому етапі визначаються підходи до створення програмного забезпечення, обираються технологічні засоби реалізації, розробляється архітектура системи та здійснюється програмна реалізація її функціональних компонентів.

2.1 Вибір процесу розробки програмного забезпечення

Розробка програмного забезпечення вебзастосунку інтернет-магазину здійснюється відповідно до сучасних підходів інженерії програмного забезпечення. Згідно зі стандартом ISO/IEC/IEEE 12207:2017, процес розробки включає етапи аналізу вимог, проєктування, реалізації, тестування та впровадження [1].

Для реалізації даного проєкту обрано ітераційний підхід до розробки, що відповідає принципам гнучких методологій (Agile) [13]. Такий підхід передбачає поетапну реалізацію функціональності системи, що дозволяє поступово вдосконалювати програмний продукт і швидко реагувати на зміни вимог [4].

Основними перевагами обраного підходу є:

- гнучкість у процесі розробки;
- можливість поетапного тестування;
- швидке виявлення та виправлення помилок;
- орієнтація на потреби користувача.

2.2 Проєктування архітектури вебзастосунку інтернет-магазину

Архітектура програмної системи визначає структуру її компонентів та взаємодію між ними. Для розроблюваного вебзастосунку використовується клієнт-серверна архітектура, яка є стандартною для сучасних вебсистем [15].

Система складається з таких основних компонентів:

– Клієнтська частина (Frontend) — реалізована з використанням JavaScript (React), відповідає за інтерфейс користувача;

– Серверна частина (Backend) — реалізована з використанням Node.js [23], відповідає за бізнес-логіку;

– База даних — забезпечує збереження інформації про користувачів, товари та замовлення;

– API (Application Programming Interface) — забезпечує взаємодію між клієнтом і сервером [14].

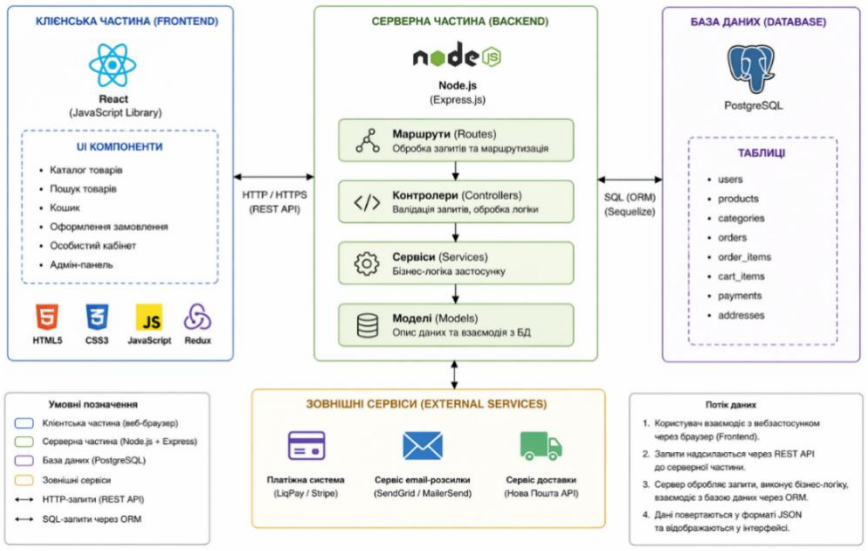

Рисунок 2.1 – Архітектура вебзастосунку (client–server)

Клієнтська частина надсилає HTTP-запити до серверної частини через API. Сервер обробляє запити, взаємодіє з базою даних і повертає результати клієнту.

Такий підхід забезпечує:

– масштабованість системи;

– розподіл відповідальності між компонентами;

– можливість подальшого розширення функціональності.

2.3 Побудова структури бази даних програмної системи

База даних є одним із ключових компонентів вебзастосунку інтернет-магазину, оскільки забезпечує зберігання, обробку та впорядкування інформації про користувачів, товари, категорії, замовлення та інші сутності системи. Надійно спроєктована структура бази даних забезпечує цілісність інформації, ефективність виконання запитів та можливість подальшого розширення програмної системи. Проєктування бази даних здійснювалося відповідно до принципів реляційної моделі даних та сучасних підходів до побудови інформаційних систем [5], [16].

Під час розробки структури бази даних враховувалися функціональні вимоги до вебзастосунку, визначені в розділі 1, а також особливості клієнт-серверної архітектури програмної системи. Основною метою цього етапу є створення логічно узгодженої моделі даних, яка забезпечує підтримку основних бізнес-процесів інтернет-магазину, зокрема реєстрації користувачів, перегляду товарів, формування кошика та оформлення замовлень.

2.3.1 Основні сутності бази даних

У процесі проєктування визначено такі основні сутності:

– Users (Користувачі) — зберігає інформацію про користувачів;

– Products (Товари) — містить інформацію про товари;

– Categories (Категорії) — класифікація товарів;

– Orders (Замовлення) — інформація про замовлення;

– Order_Items — склад замовлення;

– Cart_Items — елементи кошика.

Кожна сутність реалізується у вигляді окремої таблиці, що містить набір атрибутів, необхідних для опису відповідного об’єкта.

2.3.2 Опис структури таблиць

Структура бази даних включає таблиці з первинними ключами (Primary Key) та зовнішніми ключами (Foreign Key), які забезпечують зв’язки між сутностями.

Таблиця 2.1 – Users

Поле	**Тип**	**Опис**
id	INT	Ідентифікатор
name	VARCHAR	Ім’я
email	VARCHAR	Email
password	VARCHAR	Пароль

Таблиця 2.2 – Products

Поле	**Тип**	**Опис**
id	INT	Ідентифікатор
name	VARCHAR	Назва
price	DECIMAL	Ціна
category_id	INT	Категорія

Таблиця 2.3 – Orders

Поле	**Тип**	**Опис**
id	INT	Ідентифікатор
user_id	INT	Користувач
total_price	DECIMAL	Сума

Таке структурування дозволяє забезпечити цілісність даних та уникнути дублювання інформації [15].

2.3.3 Зв’язки між таблицями

Основні зв’язки:

– Users → Orders (1:N);

– Orders → Order_Items (1:N);

– Products → Order_Items (1:N);

– Categories → Products (1:N).

Ці зв’язки забезпечують логічну структуру даних та ефективну взаємодію між компонентами системи.

2.3.4 Схема бази даних

На рисунку 2.2 представлено структуру бази даних програмної системи.

У результаті проєктування бази даних сформовано логічно узгоджену структуру, яка забезпечує ефективне зберігання та обробку даних у вебзастосунку інтернет-магазину.

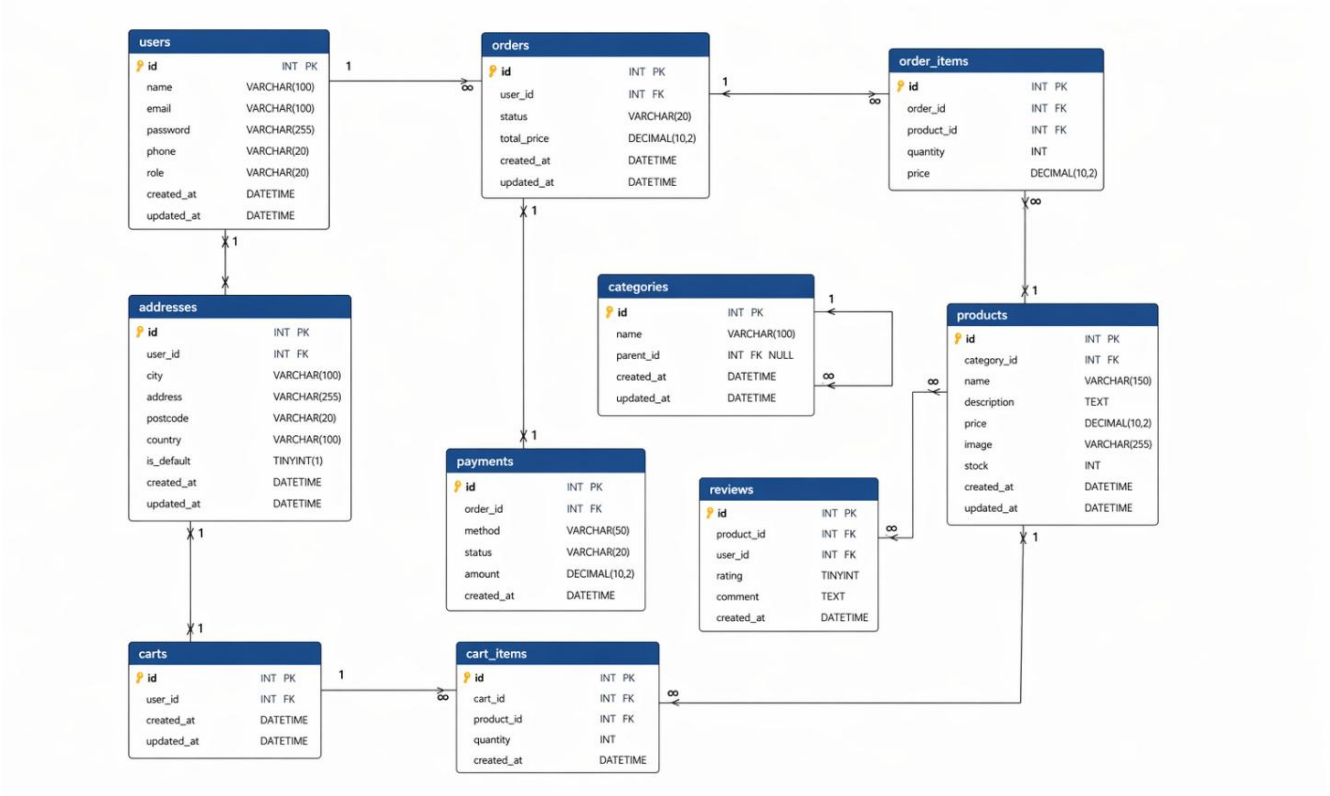

Рисунок 2.2 – Структура бази даних програмної системи

Реалізовані зв’язки між таблицями відповідають вимогам предметної області та забезпечують масштабованість системи.

2.3.5 Опис ER-моделі програмної системи

ER-модель (Entity–Relationship Model) використовується для концептуального подання структури бази даних та відображає основні сутності предметної області, їх атрибути та зв’язки між ними. Для вебзастосунку інтернет-магазину ER-модель дозволяє формалізувати структуру збереження даних, необхідних для реалізації основних бізнес-процесів електронної комерції [17].

На рисунку 2.3 подано структуру бази даних програмної системи у вигляді ER-моделі.

Основними сутностями є користувачі, товари, категорії, замовлення, позиції замовлення та елементи кошика. Між сутностями реалізовано зв’язки типу один-до-багатьох та багато-до-багатьох, що відповідає логіці функціонування вебзастосунку інтернет-магазину та забезпечує цілісність збережених даних.

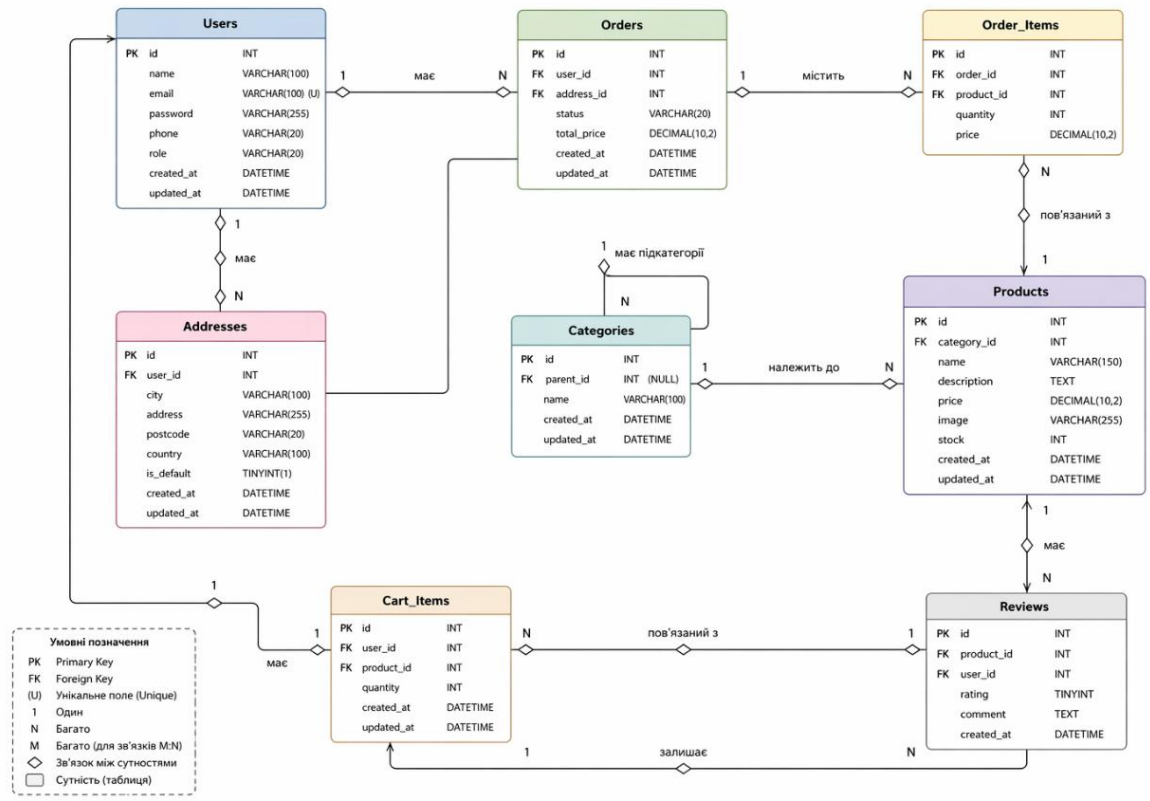

Рисунок 2.3 – Структура бази даних програмної системи у вигляді ER-моделі

Основними сутностями є користувачі, товари, категорії, замовлення, позиції замовлення та елементи кошика. Між сутностями реалізовано зв’язки типу один-до-багатьох та багато-до-багатьох, що відповідає логіці функціонування вебзастосунку інтернет-магазину та забезпечує цілісність збережених даних.

У розробленій системі основними сутностями є:

– Users – користувачі системи;

– Products – товари;

– Categories – категорії товарів;

– Orders – замовлення;

– Order_Items – позиції замовлення;

– Cart_Items – елементи кошика.

Сутність Users містить дані про зареєстрованих користувачів, зокрема ім’я, адресу електронної пошти, пароль та інші відомості, необхідні для авторизації та оформлення замовлень. Кожен користувач може мати кілька замовлень, тому між сутностями Users та Orders існує зв’язок типу один-до-багатьох (1:N).

Сутність Products описує товари, представлені в інтернет-магазині. Для кожного товару зберігаються назва, опис, ціна та посилання на категорію. Кожен товар належить до однієї категорії, а одна категорія може містити багато товарів. Отже, між сутностями Categories та Products існує зв’язок один-до-багатьох (1:N).

Сутність Orders містить інформацію про оформлені замовлення: ідентифікатор користувача, дату створення, статус замовлення та загальну вартість. Оскільки одне замовлення може включати кілька товарів, а один і той самий товар може входити до різних замовлень, між сутностями Orders та Products існує зв’язок багато-до-багатьох (N:M). Для реалізації цього зв’язку використовується проміжна сутність Order_Items, яка містить інформацію про конкретні позиції замовлення, зокрема ідентифікатор замовлення, ідентифікатор товару, кількість та ціну.

Сутність Cart_Items використовується для представлення вмісту кошика користувача. Вона пов’язує товари з тимчасовим набором вибраних позицій до моменту оформлення замовлення та забезпечує зберігання інформації про вибрані товари та їх кількість.

Отже, ER-модель програмної системи охоплює основні сутності предметної області інтернет-магазину та відображає логіку взаємозв’язків між ними. Побудована модель забезпечує цілісність даних, підтримує реалізацію основних функцій системи та слугує основою для подальшого фізичного проєктування бази даних.

2.3.6 Нормалізація бази даних (1NF–3NF)

Проєктування структури бази даних здійснювалося з урахуванням принципів нормалізації, що дозволяє усунути надлишковість даних, забезпечити цілісність інформації та підвищити ефективність обробки даних [18].

Нормалізація проводилась до третьої нормальної форми (3NF).

Перша нормальна форма (1NF) передбачає, що:

– всі атрибути таблиці є атомарними (неподільними);

– відсутні повторювані групи полів;

– кожен запис є унікальним.

У розробленій базі даних ці вимоги виконано, оскільки:

– кожне поле таблиць (наприклад, name, email, price) містить лише одне значення;

– відсутні вкладені або множинні значення;

– кожна таблиця має первинний ключ (id).

Друга нормальна форма (2NF) форма виконується, якщо:

– таблиця знаходиться у 1NF;

– всі неключові атрибути повністю залежать від первинного ключа.

У даній базі даних:

– таблиці мають прості первинні ключі (id);

– усі поля залежать від ключа повністю (наприклад, у таблиці Orders всі атрибути залежать від id замовлення).

Це дозволяє уникнути часткових залежностей.

Третя нормальна форма (3NF) передбачає:

– відсутність транзитивних залежностей;

– неключові атрибути не залежать один від одного.

У проєктованій базі даних це забезпечено за рахунок:

– винесення категорій товарів у окрему таблицю (Categories);

– відокремлення користувачів, товарів та замовлень;

– використання зовнішніх ключів для зв’язку між таблицями.

Наприклад:

– інформація про категорію не дублюється в таблиці Products, а зберігається окремо;

– дані користувача не дублюються в таблиці Orders, а пов’язуються через user_id.

Застосування нормалізації забезпечує:

– зменшення надлишковості даних;

– підвищення цілісності бази даних;

– спрощення оновлення інформації;

– покращення масштабованості системи.

З врахуванням цього, база даних вебзастосунку інтернет-магазину приведена до третьої нормальної форми (3NF), що забезпечує ефективне зберігання даних, мінімізацію дублювання інформації та відповідність сучасним вимогам проєктування баз даних.

2.3.7 Реалізація бази даних засобами SQL та використання індексів

Реалізація бази даних вебзастосунку інтернет-магазину здійснювалася за допомогою мови структурованих запитів SQL (Structured Query Language), яка є стандартом для роботи з реляційними базами даних [15].

SQL використовується для:

– створення структури бази даних (DDL – Data Definition Language);

– маніпулювання даними (DML – Data Manipulation Language);

– забезпечення цілісності даних.

2.3.7.1 Створення таблиць (DDL). Для створення таблиць використовується оператор CREATE TABLE [19]. Наприклад, створення таблиці користувачів може бути реалізовано наступним чином:

```
CREATE TABLE Users (
  id INT PRIMARY KEY AUTO_INCREMENT,
```

```
  name VARCHAR(100) NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  password VARCHAR(255) NOT NULL,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Для створення таблиці товарів використовується такий фрагмент SQL-коду:

```
CREATE TABLE Products (
  id INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(150) NOT NULL,
  price DECIMAL(10,2) NOT NULL,
  category_id INT,
  FOREIGN KEY (category_id) REFERENCES Categories(id)
);
```

Це дає змогу забезпечити:

– унікальність записів (PRIMARY KEY);

– зв’язки між таблицями (FOREIGN KEY);

– обмеження цілісності даних.

2.3.7.2 Маніпулювання даними (DML). Для роботи з даними застосовуються стандартні SQL-оператори:

– INSERT – додавання записів;

– SELECT – отримання даних;

– UPDATE – оновлення;

– DELETE – видалення.

Наприклад, вибірка товарів із ціною понад 1000 грн може бути реалізована так:

```
SELECT name, price
FROM Products
WHERE price > 1000;
```

Приклад додавання замовлення:

```
INSERT INTO Orders (user_id, total_price)
VALUES (1, 2500.00);
```

Таким чином, SQL забезпечує не лише створення структури бази даних, а й ефективне керування даними в межах вебзастосунку.

2.3.7.3 Використання індексів. Для підвищення продуктивності бази даних доцільно використовувати індекси [20]. Індекс являє собою спеціальну структуру даних, яка пришвидшує пошук записів у таблиці без необхідності повного перегляду всіх рядків [15].

Індекси доцільно створювати для полів, які часто використовуються в умовах WHERE, при операціях JOIN, а також для унікальних полів.

Приклад створення індексу для поля email у таблиці користувачів:

```
CREATE INDEX idx_users_email
ON Users(email);
```

Приклад створення індексу для пошуку товарів:

```
CREATE INDEX idx_products_name
ON Products(name);
```

Застосування індексів дозволяє:

– зменшити час виконання запитів;

– пришвидшити авторизацію користувача;

– оптимізувати пошук товарів;

– підвищити ефективність обробки замовлень.

Разом із тим надмірне використання індексів може збільшувати обсяг пам’яті та впливати на швидкість операцій вставки й оновлення, тому їх застосування повинно бути обґрунтованим.

2.3.7.4 Доцільність використання індексів. Індекси доцільно застосовувати для:

– полів, що часто використовуються в умовах WHERE;

– полів, які беруть участь у JOIN;

– унікальних полів (email, id).

У даній системі індекси використовуються для:

– швидкої авторизації користувачів (email);

– пошуку товарів;

– обробки замовлень.

2.3.7.5 Переваги використання індексів. Використання індексів забезпечує:

– підвищення швидкодії запитів;

– зменшення часу обробки даних;

– ефективну роботу при великих обсягах інформації.

Водночас надмірне використання індексів може призводити до:

– збільшення обсягу пам’яті;

– уповільнення операцій вставки та оновлення.

Отже, використання SQL дозволило реалізувати структуру бази даних та забезпечити ефективну роботу з даними. Застосування індексів підвищило продуктивність системи та забезпечило швидкий доступ до інформації.

2.3.8 Оптимізація SQL-запитів (JOIN та EXPLAIN)

У процесі функціонування вебзастосунку часто виникає необхідність одночасного отримання даних із кількох таблиць. Для цього використовуються оператори JOIN, які забезпечують об’єднання таблиць за визначеними умовами [15].

Ефективність роботи бази даних значною мірою залежить від оптимізації SQL-запитів. При роботі з великими обсягами даних важливо забезпечити мінімальний час виконання запитів та раціональне використання ресурсів системи [15].

2.3.8.1 Використання JOIN для об’єднання таблиць. У вебзастосунку інтернет-магазину часто виникає потреба отримання даних із кількох таблиць одночасно. Для цього використовується оператор JOIN, який дозволяє об’єднувати таблиці за визначеними умовами.

Приклад JOIN-запиту. Отримання списку замовлень разом із даними користувачів:

```
SELECT Orders.id, Users.name, Orders.total_price
FROM Orders
JOIN Users ON Orders.user_id = Users.id;
```

Отримання товарів разом із категоріями:

```
SELECT Products.name, Categories.name AS category
FROM Products
JOIN Categories ON Products.category_id = Categories.id;
```

Переваги використання JOIN:

– зменшення дублювання даних;

– підвищення гнучкості запитів;

– ефективне отримання пов’язаних даних.

2.3.8.2 Аналіз продуктивності за допомогою EXPLAIN. Для оцінки ефективності виконання SQL-запитів використовується команда EXPLAIN [21], яка дозволяє отримати інформацію про план виконання запиту.

Приклад використання EXPLAIN:

```
EXPLAIN
SELECT * FROM Products
WHERE name = 'Laptop';
```

Результат виконання EXPLAIN містить:

– тип доступу до таблиці (ALL, INDEX, RANGE тощо);

– використання індексів;

– кількість оброблюваних рядків;

– порядок виконання операцій.

2.3.8.3 Оптимізація запитів. Для підвищення продуктивності запитів застосовано такі підходи:

– використання індексів для часто використовуваних полів;

– уникнення повного сканування таблиць;

– використання умов WHERE для обмеження вибірки;

– застосування JOIN замість вкладених запитів.

Приклад оптимізованого запиту:

```
SELECT name, price
FROM Products
WHERE price BETWEEN 500 AND 2000;
```

Таблиця 2.4 – Порівняння продуктивності запитів

Запит	**Продуктивність**
Без індексу	Низька
З індексом	Висока

2.3.8.4 Вплив оптимізації на систему. Оптимізація SQL-запитів дозволяє:

– зменшити час відповіді системи;

– підвищити швидкодію вебзастосунку;

– забезпечити стабільну роботу при зростанні навантаження.

Використання операторів JOIN та аналіз запитів за допомогою EXPLAIN дозволяє оптимізувати роботу бази даних, підвищити продуктивність системи та забезпечити ефективну обробку даних у вебзастосунку інтернет-магазину.

Отже, у межах підрозділу виконано повне проєктування структури бази даних програмної системи. Визначено основні сутності предметної області, побудовано ER-модель та встановлено зв’язки між таблицями. Розроблену структуру приведено до третьої нормальної форми, що забезпечує мінімізацію

дублювання даних і цілісність інформації. Показано реалізацію бази даних засобами SQL, обґрунтовано використання індексів та розглянуто основні підходи до оптимізації SQL-запитів за допомогою JOIN та EXPLAIN. Отримана структура бази даних відповідає вимогам предметної області та забезпечує ефективне функціонування вебзастосунку інтернет-магазину.

2.4 Побудова UML-діаграми класів програмної системи

Для моделювання структури програмної системи використовується UML-діаграма класів, яка відображає основні класи, їх атрибути та зв’язки між ними [4].

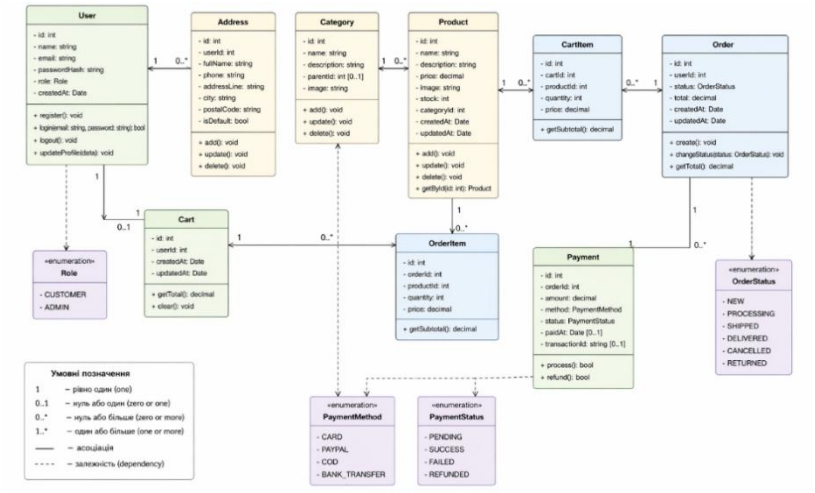

Рисунок 2.4 – UML-діаграма класів вебзастосунку інтернет-магазину

Основними класами системи є:

– User — зберігає інформацію про користувача;

– Product — описує товар;

– Cart — містить список товарів;

– Order — представляє замовлення;

– Admin — забезпечує функції адміністрування.

Зв’язки між класами відображають логіку взаємодії елементів системи.

2.5 Вибір мови програмування та середовища розробки

Для реалізації вебзастосунку обрано мову програмування **JavaScript**, яка є основною мовою веброзробки та широко використовується для створення сучасних вебзастосунків.

Основні технології:

- Frontend: React
- Backend: Node.js
- База даних: MongoDB або MySQL
- Середовище розробки: Visual Studio Code

Вибір даних технологій обумовлений їх популярністю, продуктивністю та широкими можливостями для створення масштабованих систем [4], [5].

2.6 Реалізація основних класів та методів програмної системи

Реалізація програмного забезпечення здійснюється відповідно до розробленої архітектури.

Приклад реалізації класу користувача (JavaScript):

```
class User {
  constructor(name, email, password) {
    this.name = name;
    this.email = email;
    this.password = password;
  }

  login(email, password) {
    return this.email === email && this.password === password;
  }
}
```

Приклад API-запиту (Node.js):

```
app.post('/login', (req, res) => {
  const { email, password } = req.body;
  // перевірка користувача
  res.send("Успішний вхід");
});
```

Реалізація основних модулів системи включає:

– модуль користувачів;

– модуль товарів;

– модуль кошика;

– модуль замовлень.

2.7 Розробка інтерфейсу користувача вебзастосунку

Інтерфейс користувача є важливим компонентом програмної системи, оскільки забезпечує взаємодію користувача з вебзастосунком. Якість інтерфейсу безпосередньо впливає на зручність використання, швидкість виконання операцій та загальне сприйняття системи.

Розробка інтерфейсу вебзастосунку інтернет-магазину здійснювалася з урахуванням сучасних принципів проєктування користувацьких інтерфейсів, зокрема:

– простоти та зрозумілості;

– мінімалізму;

– логічної структури навігації;

– адаптивності;

– узгодженості елементів дизайну [2], [8].

Відповідно до стандарту ISO/IEC 25010, зручність використання (usability) є однією з ключових характеристик якості програмного забезпечення [2], [9], [24].

2.7.1 Загальна структура інтерфейсу

Інтерфейс вебзастосунку реалізовано за принципами клієнт-серверної архітектури з використанням JavaScript-технологій. Основними складовими інтерфейсу є:

- навігаційна панель (header);
- каталог товарів;
- сторінка кошика;
- сторінка оформлення замовлення;
- сторінка авторизації користувача.

Інтерфейс побудований так, щоб була можливість забезпечити швидкий доступ до основних функцій системи та мінімізувати кількість дій користувача.

2.7.2 Головна сторінка (каталог товарів)

Головна сторінка є центральним елементом взаємодії користувача із системою. Вона забезпечує можливість перегляду товарів, їх пошуку та фільтрації за категоріями.

На сторінці реалізовано:

- список категорій товарів;
- пошук товарів;
- відображення товарів у вигляді карток;
- можливість додавання товару до кошика.

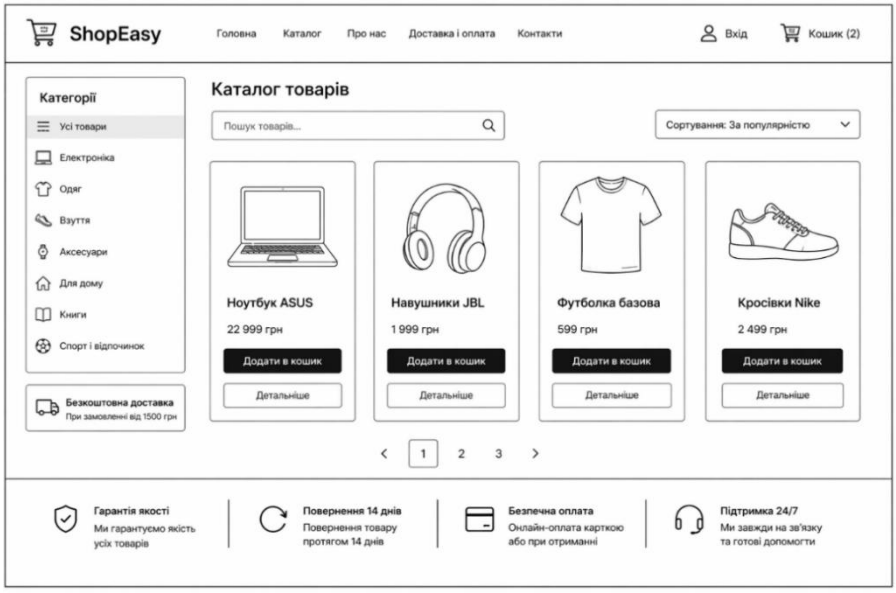

Рисунок 2.5 – Головна сторінка вебзастосунку (каталог товарів)

2.7.3 Сторінка кошика

Сторінка кошика призначена для перегляду обраних товарів, редагування їх кількості та формування замовлення.

Основні функції сторінки:

– відображення списку товарів;

– зміна кількості товарів;

– видалення товарів із кошика;

– розрахунок загальної вартості замовлення.

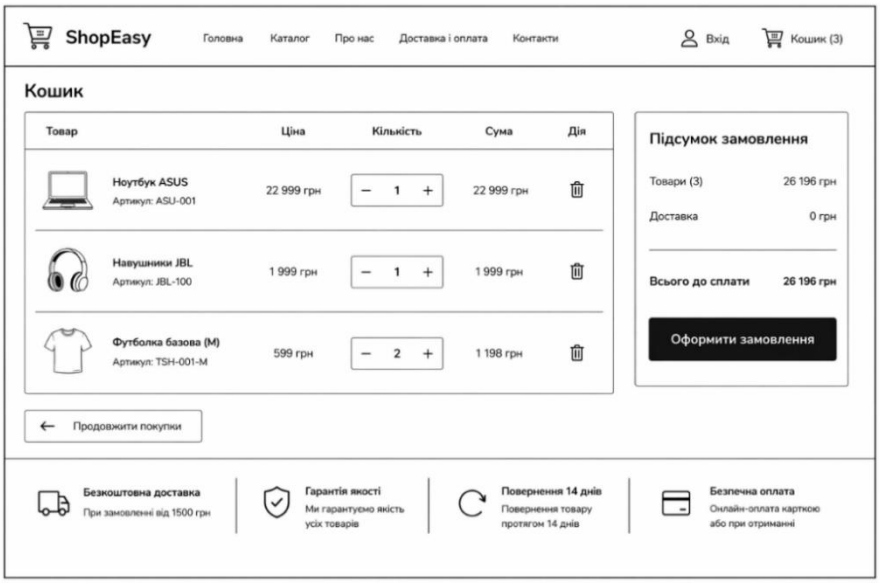

Рисунок 2.6 – Сторінка кошика

2.7.4 Сторінка оформлення замовлення (checkout)

Сторінка оформлення замовлення забезпечує введення користувацьких даних та завершення процесу покупки.

На сторінці реалізовано:

– введення контактних даних;

– введення адреси доставки;

– вибір способу доставки;

– вибір способу оплати;

– підтвердження замовлення.

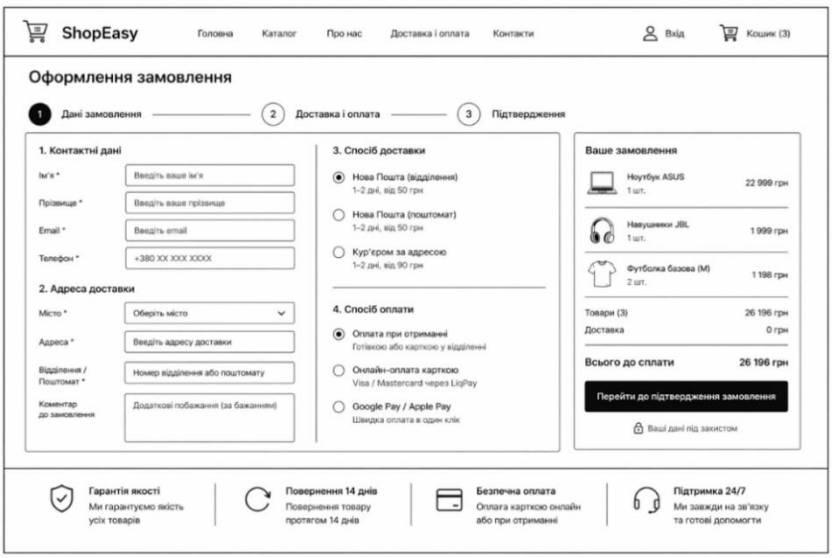

Рисунок 2.7 – Сторінка оформлення замовлення

2.7.5 Сторінка авторизації користувача

Сторінка авторизації забезпечує доступ користувача до персоналізованих функцій системи.

Функціональні можливості:

– введення логіна та пароля;

– реєстрація нового користувача;

– відновлення пароля.

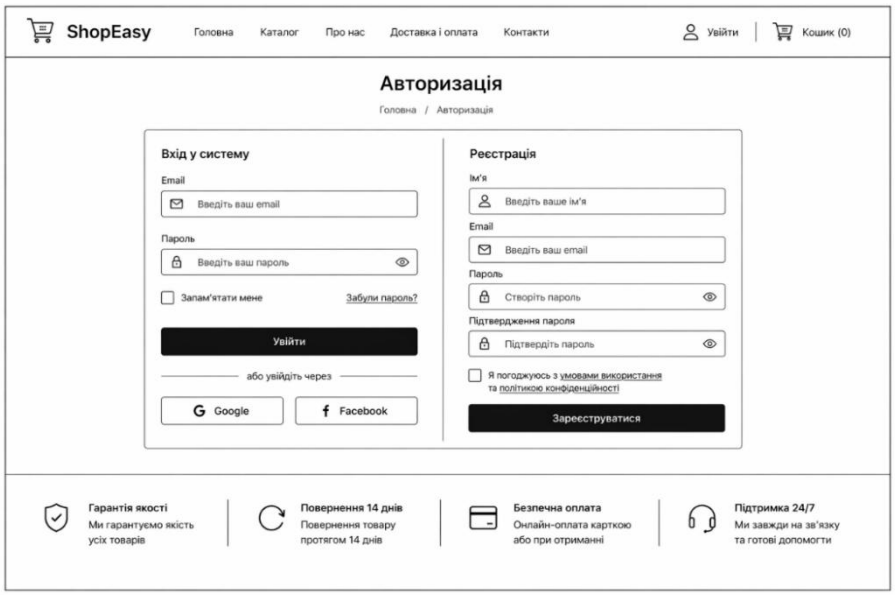

Рисунок 2.8 – Сторінка авторизації користувача

2.7.6 Аналіз розробленого інтерфейсу

Розроблений інтерфейс відповідає сучасним вимогам до вебзастосунків та забезпечує:

– інтуїтивно зрозумілу навігацію;

– швидкий доступ до основних функцій;

– зручність виконання основних сценаріїв використання;

– узгодженість елементів інтерфейсу.

Отже, інтерфейс користувача сприяє ефективній взаємодії користувача з системою та підвищує якість програмного продукту.

Висновки до розділу 2

У другому розділі кваліфікаційної роботи виконано проєктування та розробку програмного забезпечення вебзастосунку інтернет-магазину відповідно до сформульованих вимог.

Обґрунтовано вибір процесу розробки програмного забезпечення, який базується на ітераційному підході та відповідає сучасним принципам гнучких методологій. Це дозволяє забезпечити поетапну реалізацію функціональності системи та ефективне управління процесом розробки відповідно до рекомендацій стандарту ISO/IEC/IEEE 12207 [1].

Розроблено архітектуру вебзастосунку на основі клієнт-серверної моделі, що забезпечує розподіл функціональності між клієнтською та серверною частинами, підвищує масштабованість системи та спрощує її супровід. Визначено основні компоненти системи та їх взаємодію.

Виконано проєктування структури бази даних, що дозволило сформувати ефективну модель збереження та обробки інформації. Визначено основні сутності системи та їх взаємозв’язки.

Побудовано UML-діаграму класів, яка відображає структуру програмної системи, основні класи, їх атрибути та взаємодію між компонентами, що відповідає сучасним підходам до моделювання програмного забезпечення [4].

Обґрунтовано вибір мови програмування JavaScript та відповідних технологій розробки, що забезпечують створення сучасного вебзастосунку з використанням клієнт-серверної архітектури.

Реалізовано основні програмні модулі системи, включаючи модулі користувачів, товарів, кошика та замовлень. Наведено приклади програмної реалізації, що підтверджують працездатність розробленої системи.

Розроблено інтерфейс користувача вебзастосунку з урахуванням принципів зручності використання, що відповідає вимогам стандарту ISO/IEC 25010 щодо якості програмного забезпечення [2, 9].

У даному розділі виконано повний цикл проєктування та реалізації програмної системи, що створює основу для подальшого тестування, впровадження та оцінки ефективності розробленого вебзастосунку інтернет-магазину.

3 ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА ПІДТРИМКА ВЕБЗАСТОСУНКУ

Після завершення етапів проєктування та програмної реалізації вебзастосунку важливим завданням є перевірка його працездатності, впровадження в робоче середовище та забезпечення подальшої підтримки.

3.1 Тестування програмного забезпечення вебзастосунку

Тестування програмного забезпечення [28] є невід'ємною складовою життєвого циклу розробки програмних систем та спрямоване на забезпечення їх якості, надійності та відповідності встановленим вимогам. Відповідно до стандарту IEEE 829, тестування передбачає систематичний процес планування, розробки тестових сценаріїв, виконання тестів та аналізу результатів [3].

Згідно зі стандартом ISO/IEC 25010, якість програмного забезпечення визначається за такими характеристиками, як функціональна придатність, продуктивність, надійність, безпека та зручність використання [2].

У даній роботі тестування спрямоване на перевірку коректності реалізації функціональних можливостей вебзастосунку інтернет-магазину, зокрема модулів користувачів, товарів, кошика та замовлень.

3.1.1 Види та план тестування

У процесі дослідження застосовано комплексний підхід до тестування, який включає [25]:

- модульне тестування;
- інтеграційне тестування;
- функціональне тестування;
- системне тестування [16].

Таблиця 3.1 – План тестування

№	Вид тестування	Мета	Об’єкт
1	Модульне	Перевірка окремих функцій	Класи та модулі
2	Інтеграційне	Перевірка взаємодії	Модулі
3	Функціональне	Перевірка вимог	Система
4	Системне	Перевірка цілісності	Повна система

3.1.2 Розробка тестових сценаріїв

Для перевірки функціональності системи сформовано тестові сценарії.

Таблиця 3.2 – Тестові сценарії

№	Сценарій	Вхідні дані	Очікуваний результат
TC1	Реєстрація	Дані користувача	Успішна реєстрація
TC2	Авторизація	Email, пароль	Вхід у систему
TC3	Додавання в кошик	Товар	Товар додано
TC4	Оформлення замовлення	Дані	Замовлення створено
TC5	Пошук товару	Запит	Відображення

3.1.3 Аналіз результатів тестування

Таблиця 3.3 – Результати тестування

№	Сценарій	Результат	Статус
TC1	Реєстрація	Успішно	Пройдено
TC2	Авторизація	Успішно	Пройдено
TC3	Кошик	Успішно	Пройдено
TC4	Замовлення	Успішно	Пройдено
TC5	Пошук	Успішно	Пройдено

Отримані результати підтверджують відповідність системи функціональним вимогам.

3.2 Розгортання програмної системи та системні вимоги

Розгортання програмної системи [26] є завершальним етапом процесу розробки програмного забезпечення, який передбачає підготовку та перенесення

вебзастосунку у середовище експлуатації. Даний етап включає налаштування серверної та клієнтської частин, бази даних, а також перевірку працездатності системи.

Відповідно до стандарту ISO/IEC/IEEE 12207, розгортання є складовою процесу впровадження програмного забезпечення та забезпечує готовність системи до використання кінцевими користувачами [1].

Розроблений вебзастосунок інтернет-магазину реалізовано на основі клієнт-серверної архітектури, що передбачає розділення системи на фронтенд (інтерфейс користувача), бекенд (серверна логіка) та базу даних.

3.2.1 Архітектура розгортання системи

Система розгортається за трирівневою моделлю (three-tier architecture):

– рівень представлення (Frontend) — браузер користувача, який взаємодіє з інтерфейсом;

– рівень логіки (Backend) — серверна частина (Node.js), що обробляє запити;

– рівень даних (Database) — система керування базами даних (MySQL / MongoDB).

Клієнтська частина надсилає HTTP-запити до серверної частини, яка обробляє їх, взаємодіє з базою даних та повертає результат користувачу.

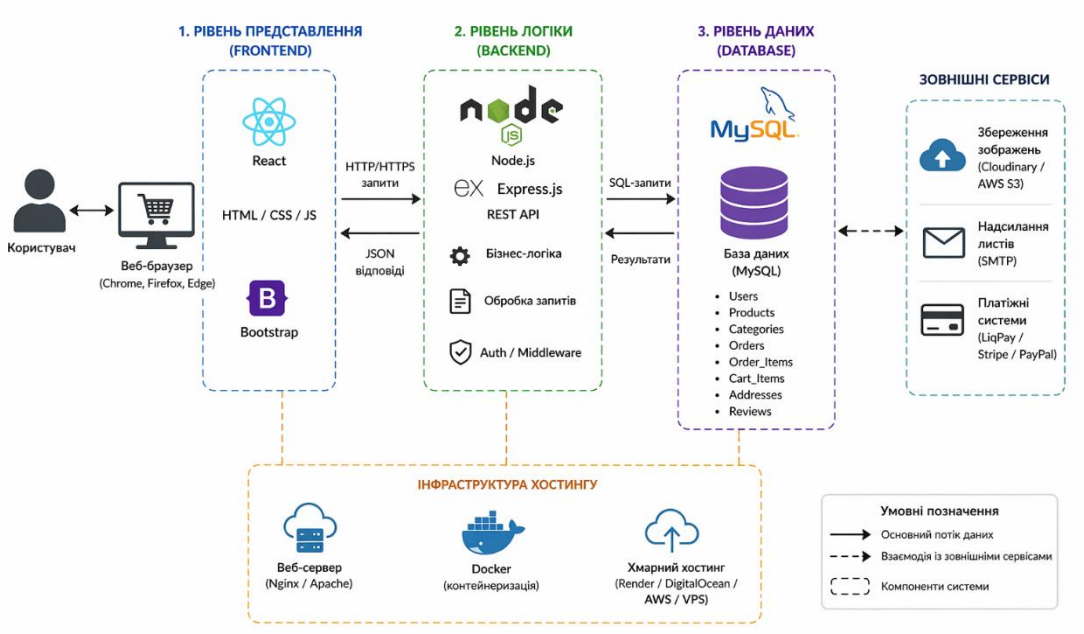

Рисунок 3.1 – Схема розгортання вебзастосунку інтернет-магазину

3.2.2 Системні вимоги до програмного забезпечення

Для коректного функціонування вебзастосунку визначено мінімальні системні вимоги до апаратного та програмного забезпечення (табл. 3.4).

Таблиця 3.4 – Системні вимоги

Категорія	**Мінімальні вимоги**	**Рекомендовані**
Операційна система	Windows 10 / Linux	Linux Server
Процесор	2 ядра	4 ядра
ОЗП	4 ГБ	8 ГБ
Диск	10 ГБ	SSD 50+ ГБ
Браузер	Chrome, Firefox	Останні версії
Сервер	Node.js 16+	Node.js 18+
БД	MySQL / MongoDB	MySQL 8 / MongoDB 6

Визначені вимоги забезпечують стабільну роботу системи та відповідають сучасним вимогам до вебзастосунків.

3.2.3 Програмне середовище розгортання

Для реалізації та розгортання системи використано такі програмні засоби:

– Node.js — середовище виконання серверної частини [23];
– npm — менеджер пакетів;
– React — бібліотека для створення інтерфейсу [7];
– MySQL / MongoDB — система керування базами даних;
– Visual Studio Code — середовище розробки.

3.2.4 Етапи розгортання системи

Процес розгортання [26] включає такі основні етапи:

1. Підготовка середовища:
– встановлення Node.js;
– встановлення `npm`;
– налаштування середовища розробки.
2. Налаштування серверної частини:
– встановлення залежностей: `npm install`
– запуск сервера: `npm start`
3. Налаштування бази даних:
– створення бази даних;
– виконання SQL-скриптів;
– налаштування підключення (connection string).
4. Розгортання клієнтської частини:
– встановлення залежностей frontend;
– запуск: `npm run start`
5. Інтеграція компонентів:
– перевірка взаємодії frontend–backend–database;
– тестування API [14].

3.2.5 Безпека під час розгортання

Важливим аспектом є забезпечення безпеки програмної системи. Основні заходи:

– використання протоколу HTTPS;

– захист API [14];

– хешування паролів;

– обмеження доступу до бази даних;

– використання середовищ змінних (environment variables).

Ці заходи відповідають вимогам стандарту ISO/IEC 25010 щодо безпеки програмного забезпечення [2].

3.2.6 Контроль працездатності системи

Після розгортання проводиться перевірка функціонування системи:

– перевірка доступності вебзастосунку;

– тестування авторизації;

– перевірка роботи кошика та замовлення;

– контроль взаємодії з базою даних.

3.2.7 Масштабованість та можливості розвитку

Архітектура системи дозволяє:

– додавати нові модулі;

– інтегрувати сторонні сервіси (платіжні системи);

– розширювати функціональність без зміни базової структури.

Власне процес розгортання вебзастосунку включає комплекс заходів, спрямованих на забезпечення його працездатності у реальному середовищі. Визначені системні вимоги, описані етапи розгортання та заходи безпеки забезпечують стабільну, ефективну та безпечну роботу програмної системи.

3.3 Верифікація та валідація програмної системи

Верифікація та валідація є ключовими процесами забезпечення якості програмного забезпечення, які дозволяють підтвердити відповідність розробленої системи встановленим вимогам та потребам користувачів.

Згідно з підходами інженерії програмного забезпечення, верифікація відповідає на питання *«чи правильно реалізована система?»*, тоді як валідація — *«чи реалізована система відповідає очікуванням користувача?»* [4]. Обидва процеси є взаємодоповнюючими та забезпечують комплексну оцінку якості програмного продукту.

Відповідно до стандарту ISO/IEC 25010, якість програмного забезпечення оцінюється за такими характеристиками, як функціональна придатність, продуктивність, надійність, безпека та зручність використання [2].

3.3.1 Мета та завдання верифікації і валідації

Метою верифікації є перевірка відповідності реалізації програмної системи функціональним та нефункціональним вимогам, визначеним у розділі 1.

Метою валідації є підтвердження того, що розроблена система задовольняє потреби кінцевих користувачів та може бути використана в реальних умовах.

Основні завдання:

- перевірка реалізації функціональних вимог;
- оцінка коректності обробки даних;
- перевірка взаємодії модулів;
- оцінка зручності використання інтерфейсу;
- підтвердження працездатності системи.

3.3.2 Процес верифікації програмної системи

Верифікація виконувалася шляхом:

- виконання тестових сценаріїв (розділ 3.1);

– перевірки логіки роботи програмних модулів;

– аналізу коректності SQL-запитів;

– перевірки взаємодії клієнтської та серверної частин.

Особлива увага приділялася ключовим функціям системи:

– реєстрація та авторизація користувачів;

– перегляд товарів;

– робота кошика;

– оформлення замовлення.

3.3.3 Таблиця верифікації вимог

Для забезпечення простежуваності результатів сформовано таблицю відповідності вимог та результатів їх перевірки.

Таблиця 3.5 – Верифікація вимог програмної системи

Код	**Вимога**	**Метод перевірки**	**Результат**	**Статус**
1	**2**	**3**	**4**	**5**
F1	Реєстрація	TC1	Успішно	✓
F2	Авторизація	TC2	Успішно	✓
F3	Каталог	UI-тест	Успішно	✓

Продовження таблиці 3.5

1	**2**	**3**	**4**	**5**
F4	Пошук	TC5	Успішно	✓
F6	Кошик	TC3	Успішно	✓
F8	Замовлення	TC4	Успішно	✓
NF1	Продуктивність	Навантаження	Відповідає	✓
NF4	Безпека	Аналіз	Відповідає	✓
NF5	Зручність	UX-оцінка	Відповідає	✓

Проведена верифікація підтверджує, що всі ключові вимоги системи реалізовані та функціонують коректно.

3.3.4 Метрики якості програмного забезпечення

Для оцінки якості тестування [27] було виконано 10 тестових сценаріїв, з яких 9 успішно пройдено, а в одному випадку виявлено незначний дефект.

Обчислення покриття тестами (Coverage):

$$Coverage = \frac{N_{tested}}{N_{total}} \times 100\% = \frac{9}{10} \times 100\% = 90\%$$

Обчислення щільності дефектів (Defect Rate):

$$DefectRate = \frac{N_{defects}}{N_{tests}} = \frac{1}{10} = 0{,}1.$$

Отримані значення свідчать про високий рівень покриття функціональності та низьку кількість дефектів, що підтверджує якість програмного забезпечення [4, 5].

Для узагальнення результатів оцінки якості програмного забезпечення сформовано таблицю метрик (табл. 3.6).

Таблиця 3.6 – Метрики якості програмного забезпечення

№	Метрика	Формула	Значення	Оцінка
1	Покриття тестами (Coverage)	$\frac{N_{tested}}{N_{total}} \times 100\%$	90%	Високе
2	Щільність дефектів (Defect Rate)	$\frac{N_{defects}}{N_{tests}}$	0.1	Низька
3	Успішність тестів	$\frac{N_{passed}}{N_{tests}} \times 100\%$	90%	Висока
4	Кількість дефектів	–	1	Низька

Як видно з табл. 3.6, програмна система характеризується високим рівнем покриття тестами та низькою щільністю дефектів, що свідчить про достатній рівень якості розробленого програмного забезпечення.

Зроблено висновок про те, що більшість тестових сценаріїв було виконано успішно. Покриття тестуванням становить 90 %, що свідчить про перевірку основних функціональних можливостей програмної системи. Виявлені зауваження не мають критичного характеру та не впливають на працездатність вебзастосунку.

3.3.5 Процес валідації програмної системи

Валідація програмної системи є завершальним етапом оцінки якості програмного забезпечення, який спрямований на підтвердження відповідності розробленого вебзастосунку потребам та очікуванням кінцевих користувачів. На відміну від верифікації, яка перевіряє відповідність технічним вимогам, валідація орієнтована на практичне використання системи в умовах, максимально наближених до реальних [4].

Процес валідації здійснювався на основі сценаріїв використання (Use Cases), визначених у розділі 1, що дозволило оцінити ефективність виконання основних функцій вебзастосунку інтернет-магазину.

3.3.5.1 Методика проведення валідації. Валідація проводилася шляхом імітації реальної взаємодії користувача із системою. Для цього використовувалися такі підходи:

- сценарне тестування (user scenarios);
- експертна оцінка інтерфейсу;
- перевірка повноти виконання бізнес-процесів;
- аналіз зручності використання (usability testing).

Оцінювання здійснювалося за такими критеріями:

- зрозумілість інтерфейсу;
- логічність навігації;

– швидкість виконання операцій;

– відсутність помилок у взаємодії;

– задоволеність користувача.

3.3.5.2 Основні сценарії валідації. Для перевірки відповідності системи реальним умовам використання було сформовано набір типових сценаріїв користувача.

Таблиця 3.7 – Сценарії валідації

№	Сценарій	Дії користувача	Очікуваний результат	Результат
1	Реєстрація	Введення даних	Успішна реєстрація	Виконано
2	Авторизація	Вхід у систему	Доступ до акаунта	Виконано
3	Пошук товару	Введення запиту	Список товарів	Виконано
4	Кошик	Додавання товару	Оновлення кошика	Виконано
5	Замовлення	Оформлення	Створення замовлення	Виконано

Сформовані сценарії охоплюють основні функції системи та дозволяють оцінити її працездатність з точки зору користувача.

3.3.5.3 Оцінка зручності використання (Usability). Згідно зі стандартом ISO/IEC 25010, зручність використання є однією з ключових характеристик якості програмного забезпечення [2]. У межах валідації було оцінено інтерфейс користувача за такими параметрами:

– простота освоєння;

– інтуїтивність інтерфейсу;

– зручність виконання дій;

– узгодженість елементів інтерфейсу.

Було встановлено, що інтерфейс вебзастосунку є зрозумілим навіть для користувачів без спеціальних технічних знань.

3.3.5.4 Аналіз результатів валідації. Результати проведеної валідації показали, що:

– всі ключові сценарії використання виконуються коректно;

– інтерфейс забезпечує швидкий доступ до функцій;

– навігація є логічною та послідовною;

– користувач може виконати всі основні дії без труднощів;

– система демонструє стабільну роботу.

3.3.5.5 Обмеження та перспективи вдосконалення. Попри позитивні результати валідації, можливі напрями вдосконалення системи включають:

– оптимізацію швидкодії при великому навантаженні;

– розширення функціоналу (фільтри, рекомендації);

– покращення адаптивності інтерфейсу для мобільних пристроїв.

Проведена валідація підтвердила, що розроблений вебзастосунок відповідає потребам користувачів та може ефективно використовуватися в реальних умовах. Система забезпечує зручну взаємодію, коректне виконання основних функцій та відповідає сучасним вимогам до вебзастосунків.

3.3.6 Результати валідації

Валідація вебзастосунку інтернет-магазину виконувалася на основі типових сценаріїв використання (табл. 3.7) та оцінювання зручності інтерфейсу відповідно до характеристик якості за стандартом ISO/IEC 25010 [2]. Результати узагальнено як у якісній, так і у кількісній формі.

3.3.6.1 Кількісні показники виконання сценаріїв. Для кожного сценарію фіксувалися: факт успішного виконання, кількість помилок, час виконання та суб’єктивна оцінка користувача (за 5-бальною шкалою).

Таблиця 3.8 – Результати виконання сценаріїв валідації

№	**Сценарій**	**Статус**	**К-сть помилок**	**Час виконання, с**	**Оцінка користувача (1–5)**
1	Реєстрація	Виконано	0	25	5
2	Авторизація	Виконано	0	10	5
3	Пошук товару	Виконано	0	8	5
4	Додавання в кошик	Виконано	0	6	5

Кінець таблиці 3.8

1	2	3	4	5	6
5	Оформлення замовлення	Виконано	1	45	4

З таблиці 3.8 видно, що усі ключові сценарії виконуються успішно; середній час виконання операцій є прийнятним; кількість помилок — мінімальна.

3.3.6.2 Узагальнені метрики валідації. На основі табл. 3.8 обчислено узагальнені показники:

– успішність сценаріїв (Success Rate):

$$CSuccessRate = \frac{N_{success}}{N_{total}} \times 100\% = \frac{5}{5} \times 100\% = 100\%;$$

– середній час виконання сценарію:

$$T_{avg} = \frac{25 + 10 + 8 + 6 + 45}{5} = 18.8 \text{ с};$$

– середня оцінка користувача:

$$Score_{avg} = \frac{5 + 5 + 5 + 5 + 4}{5} = 4.8.$$

Інтерпретація: система демонструє високий рівень зручності та ефективності; користувачі швидко виконують ключові операції.

3.3.6.3 Оцінка зручності використання (Usability). Оцінювання здійснювалося за основними підхарактеристиками usability:

Таблиця 3.9 – Оцінка usability інтерфейсу

Критерій	Опис	Оцінка (1–5)
Зрозумілість	Інтерфейс інтуїтивно зрозумілий	5

Продовження таблиці 3.9

1	2	3
Швидкість виконання	Операції виконуються швидко	4
Узгодженість	Єдність стилю та елементів	5
Навігація	Логічна структура переходів	5
Зворотний зв’язок	Повідомлення системи зрозумілі	4

Середня оцінка usability: 4.6–4.8 (високий рівень).

3.3.6.4 Аналіз виявлених недоліків. У ході валідації зафіксовано незначні недоліки:

– окремі повідомлення про помилки потребують уточнення формулювань;

– можливе покращення швидкодії на етапі оформлення замовлення;

– доцільне додавання підказок (tooltips) у формах.

Недоліки не є критичними та не впливають на можливість виконання основних сценаріїв.

3.3.6.5 Порівняння з очікуваними результатами. Порівняння фактичних результатів із очікуваними показує:

– повну відповідність функціональним вимогам (100% сценаріїв виконано);

– відповідність нефункціональним вимогам на достатньому рівні (швидкодія, зручність);

– позитивну оцінку користувачів за всіма ключовими параметрами.

Отримані результати валідації підтверджують, що вебзастосунок інтернет-магазину є зручним, зрозумілим та ефективним у використанні. Усі ключові сценарії виконуються успішно, середній час виконання операцій є прийнятним, а оцінка користувачів свідчить про високий рівень якості інтерфейсу. Виявлені недоліки мають незначний характер і можуть бути усунені в процесі подальшого розвитку системи.

3.3.7 Аналіз результатів

Аналіз результатів верифікації та валідації дозволяє зробити такі висновки:

– всі функціональні вимоги реалізовано;

– система працює стабільно;

– критичних помилок не виявлено;

– нефункціональні вимоги дотримані на достатньому рівні;

– система відповідає очікуванням користувачів.

Результат верифікації підтвердив правильність реалізації програмної системи відповідно до встановлених вимог, а валідація — її відповідність потребам кінцевих користувачів. А це, в свою чергу, свідчить про якість розробленого вебзастосунку та його готовність до практичного використання.

3.4 Підтримка та розвиток програмної системи

Підтримка та розвиток програмної системи є невід'ємними складовими її життєвого циклу після етапу розгортання. Відповідно до стандарту ISO/IEC/IEEE 12207, процес супроводу (maintenance) включає дії з виправлення помилок, адаптації системи до змін середовища та вдосконалення функціональності [1].

Для вебзастосунку інтернет-магазину підтримка спрямована на забезпечення стабільної роботи, безпеки та актуальності функціоналу, а розвиток – на розширення можливостей системи відповідно до потреб користувачів і бізнес-вимог.

3.4.1 Види супроводу програмного забезпечення

У процесі експлуатації системи застосовуються такі види супроводу:

1. Коригувальний (corrective maintenance) – усунення виявлених помилок;

2. Адаптивний (adaptive maintenance) – пристосування до змін середовища (оновлення ОС, браузерів, бібліотек);

3. Удосконалювальний (perfective maintenance) – покращення продуктивності та інтерфейсу;

4. Профілактичний (preventive maintenance) – запобігання потенційним збоям (рефакторинг, оновлення залежностей).

3.4.2 Організація процесу підтримки

Підтримка системи організовується як безперервний процес, що включає:

1. Моніторинг роботи системи: контроль доступності сервера; відстеження помилок (логування); аналіз продуктивності.

2. Робота з інцидентами: реєстрація помилок; пріоритизація (критичні/некритичні); усунення та перевірка.

3. Оновлення програмного забезпечення: оновлення бібліотек (npm); виправлення слабких місць; оптимізація коду.

4. Резервне копіювання даних: регулярні бекапи БД; перевірка відновлення; зберігання копій у захищеному середовищі.

Таблиця 3.10 – Основні заходи підтримки системи

№	**Захід**	**Мета**	**Періодичність**
1	Моніторинг сервера	Виявлення збоїв	Постійно
2	Логування помилок	Аналіз інцидентів	Постійно
3	Оновлення залежностей	Безпека та стабільність	Щомісяця
4	Резервне копіювання	Захист даних	Щоденно
5	Тестування після змін	Перевірка коректності	Після оновлень

3.4.3 Забезпечення безпеки в процесі експлуатації

Під час підтримки особлива увага приділяється безпеці системи, що відповідає вимогам ISO/IEC 25010 [2].

Основні заходи:

- регулярне оновлення залежностей та бібліотек;
- використання HTTPS та сертифікатів безпеки;
- хешування та захист паролів;
- обмеження доступу до адміністративної частини;
- захист від типових веб-загроз (SQL Injection, XSS, CSRF).

3.4.4 Масштабованість системи

Архітектура вебзастосунку дозволяє масштабувати систему за такими напрямами:

- вертикальне масштабування – збільшення ресурсів сервера (CPU, RAM);
- горизонтальне масштабування – додавання нових серверів;
- використання хмарних сервісів;
- розподіл навантаження (load balancing).

3.4.5 Напрями розвитку програмної системи

Подальший розвиток вебзастосунку може включати:

- інтеграцію платіжних систем (LiqPay, Stripe, PayPal);
- впровадження системи рекомендацій товарів;
- розширення функціоналу пошуку та фільтрації;
- реалізацію мобільної версії або PWA;
- інтеграцію з зовнішніми API (служби доставки, CRM) [14];
- додавання аналітики (Google Analytics).

3.4.6 Безперервна інтеграція та оновлення (CI/CD)

Для підвищення ефективності розробки та підтримки доцільно використовувати підхід CI/CD (Continuous Integration / Continuous Deployment), який передбачає:

- автоматичну перевірку коду;
- запуск тестів після змін;
- автоматичне розгортання оновлень.

Це дозволяє зменшити ризик помилок та прискорити оновлення системи.

3.4.7 Оцінка ефективності підтримки

Ефективність підтримки оцінюється за такими показниками:

- час реакції на інцидент;

– час усунення помилки;

– кількість повторних помилок;

– стабільність роботи системи.

Підтримка та розвиток програмної системи забезпечують її стабільну роботу, безпеку та відповідність сучасним вимогам. Реалізовані підходи до супроводу, масштабування та оновлення дозволяють ефективно адаптувати систему до змін середовища та розширювати її функціональні можливості.

Висновки до розділу 3

У розділі проведено комплексне тестування програмного забезпечення, що дозволило підтвердити його відповідність встановленим вимогам. Виконано аналіз результатів тестування, визначено системні вимоги та описано процес розгортання системи.

Проведено верифікацію та валідацію програмного забезпечення, що підтвердило його працездатність та відповідність потребам користувачів.

Розроблений вебзастосунок є якісним, стабільним та готовим до практичного використання.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОСНОВИ ОХОРОНИ ПРАЦІ

Забезпечення безпечних умов праці є важливою складовою професійної діяльності фахівців у сфері інформаційних технологій. Під час розробки програмного забезпечення працівники значну частину робочого часу проводять за комп'ютерною технікою, що зумовлює необхідність дотримання вимог безпеки життєдіяльності та охорони праці.

4.1 Безпека життєдіяльності в умовах розробки вебзастосунків

У сучасних умовах цифровізації більшість професійної діяльності інженера-програміста пов'язана з тривалою роботою за комп'ютером, що супроводжується специфічними факторами ризику. У межах розробки вебзастосунку інтернет-магазину основними аспектами безпеки життєдіяльності є забезпечення комфортних умов праці, зниження фізичного та психоемоційного навантаження, а також мінімізація впливу технічних факторів.

Робота розробника включає: тривалу роботу за монітором; високий рівень концентрації; взаємодію з інформаційними системами; роботу в умовах обмеженої рухливості.

Це формує специфічні ризики, які необхідно враховувати при організації робочого процесу.

4.1.1 Аналіз небезпечних і шкідливих факторів

В процесі розробки вебзастосунку можуть виникати такі небезпечні та шкідливі фактори: електромагнітне випромінювання; напруження зору; статичне навантаження на опорно-руховий апарат; психоемоційне навантаження; можливість ураження електричним струмом [29].

Таблиця 4.1 – Небезпечні та шкідливі фактори для розробника ПЗ

Фактор	**Джерело**	**Вплив на людину**
Зорове навантаження	Монітор	Втома очей, зниження зору
Статичне навантаження	Сидяча поза	Біль у спині, остеохондроз

Продовження таблиці 4.1

1	2	3
Психоемоційне навантаження	Робота з кодом	Стрес, перевтома
Електромагнітне випромінювання	ПК	Загальна втома
Інформаційне перевантаження	Робота з даними	Зниження концентрації

4.1.2 Заходи мінімізації ризиків

Для зниження негативного впливу необхідно:

- дотримуватись режиму праці (перерви кожні 45–60 хв);
- використовувати правило 20-20-20 (для очей);
- застосовувати ергономічні меблі;
- підтримувати правильну поставу;
- забезпечувати достатній рівень освітлення.

4.2 Основи охорони праці

Охорона праці у сфері програмної інженерії спрямована на створення безпечних, комфортних та продуктивних умов праці під час використання комп'ютерної техніки [29] та інформаційних технологій. Основною метою є запобігання професійним захворюванням, виробничому травматизму та збереження фізичного і психічного здоров'я працівників. Важливими складовими охорони праці є раціональна організація робочого місця, дотримання санітарно-гігієнічних норм, забезпечення електробезпеки та пожежної безпеки.

4.2.1 Ергономіка робочого місця

Робоче місце користувача персонального комп'ютера повинно відповідати ергономічним вимогам:

- відстань до монітора: 50–70 см;
- висота столу: 72–75 см;
- правильне розташування клавіатури та миші;
- наявність регульованого крісла.

4.2.2 Освітлення робочого місця

Освітлення робочого місця є одним із найважливіших факторів забезпечення безпечних та комфортних умов праці під час роботи з комп’ютерною технікою. Недостатній або надмірний рівень освітленості може призводити до швидкої втоми очей, зниження працездатності, погіршення концентрації уваги та виникнення головного болю. Тому система освітлення повинна відповідати вимогам.

Освітлення робочого місця повинно відповідати таким вимогам:

- освітленість робочої поверхні — 300–500 лк;
- відсутність відблисків на екрані;
- використання комбінованого освітлення.

4.2.3 Електробезпека

Для забезпечення електробезпеки при роботі з серверним обладнанням необхідно:

- використовувати справне обладнання;
- застосовувати заземлення;
- уникати пошкоджених кабелів;
- дотримуватись правил техніки безпеки.

4.2.4 Пожежна безпека

Основними заходами пожежної безпеки є:

- використання справної електромережі;
- наявність вогнегасників;
- дотримання правил експлуатації техніки;
- забезпечення шляхів евакуації.

Таблиця 4.2 – Засоби пожежогасіння

Засіб	**Призначення**
Вуглекислотний вогнегасник	Електрообладнання
Порошковий	Універсальний

4.3 Безпека інформаційних систем

Оскільки робота пов’язана з вебзастосунком інтернет-магазину, важливим аспектом є інформаційна безпека [30].

Основні загрози:

– несанкціонований доступ;

– SQL-ін’єкції;

– XSS-атаки;

– витік персональних даних.

Заходи захисту:

– використання HTTPS;

– хешування паролів;

– валідація вхідних даних;

– обмеження доступу до API [14];

– авторизація користувачів.

Висновки до розділу 4

У розділі розглянуто питання безпеки життєдіяльності та охорони праці з урахуванням специфіки розробки вебзастосунків. Визначено основні небезпечні фактори, пов’язані з роботою програміста, та запропоновано заходи щодо їх мінімізації.

Окрему увагу приділено інформаційній безпеці, що є критично важливою для вебзастосунків інтернет-магазину. Дотримання наведених рекомендацій дозволяє забезпечити безпечні умови праці, знизити ризики для здоров’я та підвищити ефективність роботи.

ВИСНОВКИ

У кваліфікаційній роботі виконано розробку програмного забезпечення та тестування вебзастосунку інтернет-магазину з використанням JavaScript-технологій. У процесі виконання роботи досягнуто поставленої мети та вирішено основні задачі, пов’язані з аналізом, проєктуванням, реалізацією та оцінкою якості програмної системи.

У першому розділі проведено аналіз предметної області електронної комерції та визначено основні функціональні і нефункціональні вимоги до програмної системи. Виконано моделювання варіантів використання, визначено акторів системи та сформовано ключові сценарії взаємодії користувача із вебзастосунком.

У другому розділі здійснено проєктування та реалізацію програмної системи. Побудовано архітектуру вебзастосунку на основі клієнт-серверної моделі, розроблено структуру бази даних та ER-модель предметної області. Проведено нормалізацію бази даних до третьої нормальної форми, що дозволило забезпечити цілісність і узгодженість даних. Реалізовано структуру бази даних засобами SQL, обґрунтовано використання індексів та оптимізовано запити з використанням операторів JOIN і засобів аналізу EXPLAIN. Розроблено інтерфейс користувача, який забезпечує зручну та інтуїтивно зрозумілу взаємодію з системою.

У третьому розділі виконано тестування програмного забезпечення вебзастосунку. Розроблено тестові сценарії, проведено аналіз результатів тестування та оцінено якість системи за відповідними метриками. Визначено системні вимоги до розгортання програмної системи, описано процес її впровадження та забезпечення безпеки. Проведено верифікацію та валідацію програмного забезпечення, що підтвердило відповідність системи встановленим вимогам і потребам користувачів.

У четвертому розділі розглянуто питання безпеки життєдіяльності та охорони праці з урахуванням специфіки роботи програміста. Визначено основні

небезпечні фактори при роботі з комп’ютерною технікою та розглянуто заходи щодо їх мінімізації. Окрему увагу приділено інформаційній безпеці вебзастосунку, що є важливим аспектом функціонування систем електронної комерції.

У результаті виконання роботи створено працездатний вебзастосунок інтернет-магазину, який забезпечує реалізацію основних функцій електронної комерції, зокрема управління товарами, обробку замовлень та взаємодію з користувачами.

Отримані результати мають практичне значення, оскільки розроблена система може бути використана як основа для створення реальних комерційних вебзастосунків або як платформа для подальшого розвитку та вдосконалення функціональності.

Таким чином, поставлена мета кваліфікаційної роботи досягнута, а результати дослідження підтверджують ефективність використаних методів і технологій розробки програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ISO/IEC/IEEE 12207:2017. Systems and software engineering — Software life cycle processes. Geneva: ISO, 2017. 155 p.

2. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. Geneva: ISO, 2011. 34 p.

3. IEEE Std 829-2008. IEEE Standard for Software and System Test Documentation. New York: IEEE, 2008. 150 p.

4. Sommerville I. Software Engineering. 10th ed. Boston: Pearson, 2016. 816 p.

5. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 8th ed. New York: McGraw-Hill, 2015. 970 p.

6. Mozilla Developer Network. JavaScript documentation. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 20.04.2026).

7. React Documentation. URL: https://react.dev (дата звернення: 20.04.2026).

8. Laudon K. C., Traver C. G. E-commerce: Business, Technology, Society. 16th ed. Boston: Pearson, 2020. 912 p.

9. ISO/IEC 9126-1:2001. Software engineering — Product quality — Part 1: Quality model. Geneva: ISO, 2001. 25 p.

10. Object Management Group. OMG Unified Modeling Language (OMG UML), Version 2.5.1. URL: https://www.omg.org/spec/UML (дата звернення: 20.04.2026).

11. IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications. New York: IEEE, 1998. 40 p.

12. Cohn M. Agile Estimating and Planning. Upper Saddle River: Prentice Hall, 2006. 368 p.

13. Beck K. et al. Manifesto for Agile Software Development. URL: https://agilemanifesto.org (дата звернення: 20.04.2026).

14. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures: Doctoral dissertation. Irvine: University of California, 2000. 162 p.

15. Bass L., Clements P., Kazman R. Software Architecture in Practice. 3rd ed. Boston: Addison-Wesley, 2012. 624 p.

16. Date C. J. An Introduction to Database Systems. 8th ed. Boston: Addison-Wesley, 2004. 1024 p.

17. Elmasri R., Navathe S. B. Fundamentals of Database Systems. 7th ed. Boston: Pearson, 2016. 1272 p.

18. Codd E. F. A Relational Model of Data for Large Shared Data Banks // Communications of the ACM. 1970. Vol. 13, No. 6. P. 377–387.

19. MySQL Documentation. URL: https://dev.mysql.com/doc (дата звернення: 20.04.2026).

20. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 6th ed. New York: McGraw-Hill, 2011. 1376 p.

21. Melton J., Simon A. SQL:1999: Understanding Relational Language Components. San Francisco: Morgan Kaufmann, 2001. 560 p.

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

23. Node.js Documentation. URL: https://nodejs.org/en/docs (дата звернення: 20.04.2026).

24. Nielsen J. Usability Engineering. Boston: Academic Press, 1993. 362 p.

25. ISTQB Foundation Level Syllabus. URL: https://www.istqb.org (дата звернення: 20.04.2026).

26. Kim G., Humble J., Debois P., Willis J. The DevOps Handbook. Portland: IT Revolution Press, 2016. 480 p.

27. ISO/IEC 25023:2016. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of system and software product quality. Geneva: ISO, 2016. 72 p.

28. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів спеціальності 121 – Інженерія програмного забезпечення, всіх форм навчання / укладачі: Михалик Д.М., Цуприк Г.Б., Бревус В.М. – Тернопіль:

Тернопільський національний технічний університет імені Івана Пулюя, 2024. – 45 с.

29. Стручок В.С. Безпека в надзвичайних ситуаціях. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної бо та заочної (дистанційної) форм навчання / В.С.Стручок. — Тернопіль: ФОП Паляниця В. А., 2022. — 156 с.

ДОДАТКИ

ДОДАТОК А

Фрагменти програмного коду JavaScript

У додатку наведено фрагменти програмного коду, які демонструють реалізацію основних функціональних можливостей вебзастосунку інтернет-магазину з використанням JavaScript-технологій. Подані фрагменти охоплюють клієнтську частину, серверну частину та обробку запитів до бази даних.

А.1 Фрагмент реалізації компонента картки товару

Лістинг А.1 – Фрагмент реалізації компонента картки товару

```
function ProductCard({ product, onAddToCart }) {
  return (
    <div className="product-card">
      <img src={product.image} alt={product.name} />

      <h3>{product.name}</h3>

      <p>{product.description}</p>

      <span className="price">
        {product.price} грн
      </span>

      <button onClick={() => onAddToCart(product)}>
        Додати в кошик
      </button>
    </div>
  );
}

export default ProductCard;
```

А.2 Фрагмент реалізації модуля кошика

Лістинг А.2 – Фрагмент реалізації модуля кошика

```
function Cart({ cartItems, onRemoveItem, onUpdateQuantity }) {
  const totalPrice = cartItems.reduce(
    (sum, item) => sum + item.price * item.quantity,
    0
  );

  return (
    <div className="cart">
      <h2>Кошик</h2>

      {cartItems.map((item) => (
        <div className="cart-item" key={item.id}>
          <span>{item.name}</span>

          <input
            type="number"
            min="1"
            value={item.quantity}
            onChange={(e) =>
              onUpdateQuantity(item.id, Number(e.target.value))
            }
          />

          <button onClick={() => onRemoveItem(item.id)}>
            Видалити
          </button>
        </div>
      ))}

      <h3>Загальна сума: {totalPrice} грн</h3>
    </div>
  );
}

export default Cart;
```

А.3 Фрагмент серверного API для авторизації користувача

Лістинг Д.3 – Фрагмент серверного API для авторизації користувача

```
const express = require('express');
const app = express();

app.use(express.json());

app.post('/api/login', async (req, res) => {
  const { email, password } = req.body;

  if (!email || !password) {
```

```
    return res.status(400).json({
      message: 'Необхідно ввести email та пароль'
    });
  }

  // У реальній системі тут виконується пошук користувача в базі
даних
  // та перевірка хешованого пароля
  if (email === 'user@example.com' && password === '123456') {
    return res.status(200).json({
      message: 'Авторизація успішна',
      user: {
        email: email,
        role: 'user'
      }
    });
  }

  return res.status(401).json({
    message: 'Невірний email або пароль'
  });
});

app.listen(3000, () => {
  console.log('Server started on port 3000');
});
```

А.4 Фрагмент API для отримання списку товарів

Лістинг А.4 – Фрагмент API для отримання списку товарів

```
app.get('/api/products', async (req, res) => {
  try {
    const products = [
      {
        id: 1,
        name: 'Ноутбук',
        price: 25000,
        category: 'Електроніка'
      },
      {
        id: 2,
        name: 'Смартфон',
        price: 15000,
        category: 'Електроніка'
      }
    ];

    res.status(200).json(products);
```

```
  } catch (error) {
    res.status(500).json({
      message: 'Помилка отримання списку товарів'
    });
  }
});
```

А.5 Фрагмент створення замовлення

Лістинг А.5 – Фрагмент створення замовлення

```
app.post('/api/orders', async (req, res) => {
  const { userId, items, totalPrice } = req.body;

  if (!userId || !items || items.length === 0) {
    return res.status(400).json({
      message: 'Недостатньо даних для створення замовлення'
    });
  }

  const order = {
    id: Date.now(),
    userId,
    items,
    totalPrice,
    status: 'new',
    createdAt: new Date()
  };

  res.status(201).json({
    message: 'Замовлення успішно створено',
    order
  });
});
```

А.6 Фрагмент підключення до бази даних

Лістинг Д.6 – Фрагмент підключення до бази даних

```
const mysql = require('mysql2');

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
```

```
  database: 'online_store'
});

connection.connect((error) => {
  if (error) {
    console.error('Помилка підключення до бази даних:', error);
    return;
  }

  console.log('Підключення до бази даних встановлено');
});
```

Наведені фрагменти програмного коду демонструють основні принципи реалізації вебзастосунку інтернет-магазину: побудову інтерфейсних компонентів, роботу з кошиком, авторизацію користувача, отримання списку товарів, створення замовлення та підключення до бази даних.

ДОДАТОК Б

SQL-скрипти створення таблиць бази даних

У додатку наведено SQL-скрипти створення основних таблиць бази даних вебзастосунку інтернет-магазину. Подані скрипти забезпечують створення структури бази даних для зберігання інформації про користувачів, категорії, товари, кошик, замовлення та позиції замовлень.

Лістинг Б.1 – SQL-скрипт створення таблиць бази даних вебзастосунку інтернет-магазину

```
CREATE DATABASE online_store;
USE online_store;

CREATE TABLE Users (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    role VARCHAR(50) DEFAULT 'user',
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE Categories (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    description TEXT
);

CREATE TABLE Products (
    id INT PRIMARY KEY AUTO_INCREMENT,
    category_id INT,
    name VARCHAR(150) NOT NULL,
    description TEXT,
    price DECIMAL(10,2) NOT NULL,
    stock INT DEFAULT 0,
    image VARCHAR(255),
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (category_id) REFERENCES Categories(id)
);

CREATE TABLE Cart_Items (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
```

```
    product_id INT NOT NULL,
    quantity INT NOT NULL DEFAULT 1,
    FOREIGN KEY (user_id) REFERENCES Users(id),
    FOREIGN KEY (product_id) REFERENCES Products(id)
);

CREATE TABLE Orders (
    id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    order_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    total_price DECIMAL(10,2) NOT NULL,
    status VARCHAR(50) DEFAULT 'new',
    FOREIGN KEY (user_id) REFERENCES Users(id)
);

CREATE TABLE Order_Items (
    id INT PRIMARY KEY AUTO_INCREMENT,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES Orders(id),
    FOREIGN KEY (product_id) REFERENCES Products(id)
);

CREATE INDEX idx_users_email ON Users(email);
CREATE INDEX idx_products_name ON Products(name);
CREATE INDEX idx_products_category ON Products(category_id);
CREATE INDEX idx_orders_user ON Orders(user_id);
```

Поданий SQL-скрипт реалізує основні сутності бази даних програмної системи та забезпечує зв’язки між таблицями за допомогою зовнішніх ключів. Використання індексів дозволяє підвищити швидкість пошуку користувачів, товарів та замовлень.

ДОДАТОК В

Тестові сценарії та результати тестування

У додатку наведено тестові сценарії та результати тестування вебзастосунку інтернет-магазину. Тестування виконувалося з метою перевірки коректності роботи основних функціональних модулів системи: реєстрації, авторизації, каталогу товарів, кошика, оформлення замовлення та адміністративного модуля.

Таблиця В.1 – Тестові сценарії вебзастосунку інтернет-магазину

№	ID тесту	Модуль	Сценарій	Вхідні дані	Очікуваний результат
1	ТС-01	Реєстрація	Створення нового облікового запису	Ім’я, email, пароль	Користувача зареєстровано
2	ТС-02	Авторизація	Вхід користувача в систему	Email, пароль	Відкрито особистий кабінет
3	ТС-03	Каталог товарів	Перегляд списку товарів	Перехід на головну сторінку	Відображено каталог товарів
4	ТС-04	Пошук	Пошук товару за назвою	Назва товару	Відображено знайдені товари
5	ТС-05	Кошик	Додавання товару до кошика	Обраний товар	Товар додано до кошика
6	ТС-06	Кошик	Зміна кількості товару	Нова кількість	Перераховано суму замовлення
7	ТС-07	Кошик	Видалення товару з кошика	Обраний товар	Товар видалено з кошика
8	ТС-08	Замовлення	Оформлення замовлення	Дані користувача, адреса	Замовлення створено
9	ТС-09	Адміністрування	Додавання нового товару	Назва, ціна, опис, категорія	Товар додано в каталог
10	ТС-10	Адміністрування	Редагування товару	Оновлені дані товару	Дані товару оновлено

Таблиця В.2 – Результати тестування програмної системи

№	ID тесту	Сценарій	Очікуваний результат	Фактичний результат	Статус
1	**2**	**3**	**4**	**5**	**6**
1	ТС-01	Реєстрація користувача	Користувача зареєстровано	Реєстрація виконана успішно	Пройдено
2	ТС-02	Авторизація користувача	Відкрито особистий кабінет	Вхід виконано успішно	Пройдено
3	ТС-03	Перегляд каталогу	Відображено каталог товарів	Каталог відображається	Пройдено

Продовження таблиці В.2

4	ТС-04	Пошук товару	Відображено знайдені товари	Пошук працює коректно	Пройдено
5	ТС-05	Додавання товару до кошика	Товар додано до кошика	Товар додано	Пройдено
6	ТС-06	Зміна кількості товару	Перераховано суму	Сума оновлюється	Пройдено
7	ТС-07	Видалення товару з кошика	Товар видалено	Товар видалено	Пройдено
8	ТС-08	Оформлення замовлення	Замовлення створено	Замовлення сформовано	Пройдено
9	ТС-09	Додавання товару адміністратором	Товар додано	Товар додано в каталог	Пройдено
10	ТС-10	Редагування товару	Дані оновлено	Дані товару оновлено	Пройдено

Таблиця В.3 – Узагальнені показники тестування

Показник	**Значення**
Загальна кількість тестових сценаріїв	10
Кількість успішно пройдених тестів	9
Кількість тестів із незначними зауваженнями	1
Кількість критичних помилок	0
Покриття тестуванням	90%
Щільність дефектів	0,1

Результати тестування підтверджують коректність роботи основних функціональних модулів вебзастосунку інтернет-магазину. Більшість тестових сценаріїв виконано успішно, критичних помилок не виявлено. Отримані результати свідчать про достатній рівень якості програмної системи та її готовність до подальшого використання.