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

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Створення веб-магазину спортивних добавок із використанням HTML, CSS, MySQL та JS

Виконав: студент IV курсу, групи СН-42
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Павлюх П.Т.
(підпис) (прізвище та ініціали)

Керівник _____ Фриз М.Є.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

Тернопіль
2025

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

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_27_» __січня__ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)
за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)
Студенту Павлюху Павлові Тарасовичу
(прізвище, ім'я, по батькові)
1. Тема роботи Створення веб-магазину спортивних добавок із використанням HTML, CSS, MySQL та JS.

Керівник роботи Фриз Михайло Євгенович, канд. техн. наук, доцент, доценто кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «*27*» *січня* 2025 року № *4/7-444*
2. Термін подання студентом завершеної роботи 26 червня 2025 р.
3. Вихідні дані до роботи Підстава для розробки – Виконання кваліфікаційної роботи бакалавра. Призначення розробки – Програма для створення повнофункціонального веб-магазину спортивних добавок для демонстрації, продажу та управління товарами, із зручним інтерфейсом для користувачів та адміністративною панеллю. Вхідні дані – Запити користувачів (URL сторінок, фільтри, дані кошика, форми реєстрації/входу, замовлення, повідомлення) та дані адміністратора (вхід, продукти, статуси). Вихідні дані – Відображення веб-сторінок магазину (головна, продукти, кошик, реєстрація, контакти, мої замовлення), динамічні списки товарів, вміст кошика, підтвердження дій, інформація про замовлення, таблиці продуктів/замовлень/повідомлень в адмін-панелі та системні повідомлення.

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка завдання. 2. Проектна частина: 2.1. Архітектура веб-додатку. 2.2. Проектування бази даних. 2.3. Проектування API. 2.4. Проектування користувацького інтерфейсу (UI/UX). 3. Реалізація програми: 3.1. Реалізація клієнтської частини (Frontend). 3.2. Реалізація серверної частини (Backend). 3.3. Тестування та налагодження. 4. Висновки. Список використаної літератури. Додатки.

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	к.т.н. доц. Сенчишин В.С.	.2025	2025

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	28.01.2025	*Виконано*
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	01.02.2025-11.02.2025	*Виконано*
3.	Виконання дослідження щодо вибору технологій (HTML, CSS, MySQL, JS, Node.js) та загального проєктування архітектури системи веб-магазину.	12.02.2025-23.02.2025	*Виконано*
4.	Оформлення розділу «Аналіз предметної області та постановка завдання»	24.02.2025-03.03.2025	*Виконано*
5.	Оформлення розділу «Проектна частина»	04.03.2025-28.03.2025	*Виконано*
6.	Оформлення розділу «Практична частина»	29.03.2025-16.04.2025	*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	17.04.2025-25.04.2025	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	26.04.2025-02.05.2025	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	03.05.2025-10.05.2025	*Виконано*
10.	Нормоконтроль	.06.2025- .06.2025	*Виконано*
11.	Перевірка на плагіат	16.06.2025	*Виконано*
12.	Попередній захист кваліфікаційної роботи	.06.2025	*Виконано*
13.	Захист кваліфікаційної роботи	.06.2025	

Студент _____ Павлюх П. Т.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Фриз М. Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Створення веб-магазину спортивних добавок із використанням HTML, CSS, MySQL та JS // Кваліфікаційна робота освітнього рівня «Бакалавр» // Павлюх Павло Тарасович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-42 // Тернопіль, 2025 // С. 60, рис. – 12, презент.сторінок – 12, додат. – 51, бібліогр. – 41.

Ключові слова: веб-магазин, спортивні добавки, HTML, CSS, JavaScript, Node.js, MySQL, електронна комерція.

Кваліфікаційна робота присвячена дослідженню та практичній реалізації веб-магазину спортивних добавок. В першому розділі кваліфікаційної роботи описано актуальність теми, проведено аналіз ринку онлайн-продажів спортивного харчування та функціоналу конкурентних веб-магазинів. Висвітлено основні функціональні та нефункціональні вимоги до системи. Розглянуто обґрунтування вибору технологічного стеку для реалізації проекту.

В другому розділі кваліфікаційної роботи детально розглянуто архітектуру веб-додатку, включаючи клієнт-серверну модель та схему взаємодії компонентів. Досліджено проектування бази даних MySQL, представлено ER-діаграму та опис таблиць. Подано проектування RESTful API з прикладами запитів та відповідей.

В третьому розділі кваліфікаційної роботи описано практичну реалізацію клієнтської частини (HTML-структура, CSS-стилі, JavaScript-логіка) та серверної частини (налаштування Node.js/Express.js, взаємодія з MySQL, реалізація API-маршрутів). Проаналізовано застосовані заходи безпеки. Проведено тестування та налагодження розробленої системи, включаючи вирішення виявлених помилок. Об'єкт дослідження: процеси електронної комерції у сфері продажу спортивних добавок. Предмет дослідження: методи та засоби створення веб-додатків для реалізації функціоналу онлайн-магазину.

ANNOTATION

Creation of an Online Store for Sports Supplements Using HTML, CSS, MySQL, and JS // Qualification work of the educational level «Bachelor» // Pavliukh Pavlo // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-42 // Ternopil, 2025 // P. 60, fig. – 12, present.pages – 12, annexes. – 51, references – 41.

Keywords: web store, sports supplements, HTML, CSS, JavaScript, Node.js, MySQL, e-commerce.

The qualification work is dedicated to the research and practical implementation of a web store for sports supplements. The first chapter of the qualification work describes the relevance of the topic, analyzes the online sales market for sports nutrition and the functionality of competing web stores. It highlights the main functional and non-functional requirements for the system. The justification for choosing the technology stack for project implementation is reviewed.

The second chapter of the qualification work thoroughly examines the web application architecture, including the client-server model and the component interaction scheme. The design of the MySQL database is explored, presenting the ER-diagram and table descriptions. The design of the RESTful API with examples of requests and responses is provided.

The third chapter of the qualification work describes the practical implementation of the client-side (HTML structure, CSS styles, JavaScript logic) and the server-side (Node.js/Express.js setup, MySQL interaction, API route implementation). The security measures applied are analyzed. Testing and debugging of the developed system, including the resolution of identified errors, have been conducted. Research object: e-commerce processes in the field of sports supplement sales. Research subject: methods and tools for creating web applications to implement online store functionality.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ТЕРМІНІВ

API (англ. Application Programming Interface) – Інтерфейс прикладного програмування.

CRUD (англ. Create, Read, Update, Delete) – Створити, прочитати, оновити, видалити (основні операції в базах даних).

CSS (англ. Cascading Style Sheets) – Каскадні таблиці стилів.

DB (англ. Database) – База даних.

HTML (англ. HyperText Markup Language) – Мова гіпертекстової розмітки.

HTTP (англ. HyperText Transfer Protocol) – Протокол передачі гіпертексту.

JS (англ. JavaScript) – Мова програмування JavaScript.

JSON (англ. JavaScript Object Notation) – Формат обміну даними JavaScript.

MySQL – Система управління реляційними базами даних.

Node.js – Кроссплатформове середовище виконання JavaScript.

NPM (англ. Node Package Manager) – Менеджер пакетів для Node.js.

REST (англ. Representational State Transfer) – Архітектурний стиль взаємодії розподілених систем.

SQL (англ. Structured Query Language) – Мова структурованих запитів.

UI (англ. User Interface) – Користувацький інтерфейс.

URL (англ. Uniform Resource Locator) – Уніфікований покажчик ресурсу.

UX (англ. User Experience) – Досвід користувача.

БД – База даних.

СУБД – Система управління базами даних.

Фронтенд – Клієнтська частина веб-додатку.

Бекенд – Серверна частина веб-додатку.

ЗМІСТ

ВСТУП

Актуальність. Сучасний світ характеризується стрімким розвитком інформаційних технологій, що трансформує всі сфери життя, включаючи комерцію. Електронна торгівля, або e-commerce, стала невід'ємною частиною глобальної економіки, пропонуючи зручні та ефективні механізми для купівлі-продажу товарів та послуг. Особливо актуальним є створення спеціалізованих онлайн-платформ, які задовольняють потреби конкретних ніш ринку. У цьому контексті, веб-магазини спортивних добавок набувають значної популярності, оскільки все більше людей приділяють увагу здоровому способу життя, фізичній активності та оптимізації спортивних досягнень за допомогою якісного харчування та добавок. Актуальність теми зумовлена необхідністю розробки сучасних, функціональних та безпечних інструментів для задоволення зростаючого попиту на спортивне харчування в онлайн-середовищі, забезпечуючи зручність для користувачів та ефективність для бізнесу.

Метою даної кваліфікаційної роботи є розробка та реалізація функціонального веб-магазину спортивних добавок, який забезпечуватиме повний цикл взаємодії з клієнтами – від перегляду каталогу до оформлення замовлення та зворотного зв'язку, а також надасть адміністративні інструменти для управління контентом та замовленнями. Для досягнення поставленої мети було визначено наступні завдання: провести аналіз предметної області та існуючих рішень; визначити функціональні та нефункціональні вимоги до системи; спроектувати архітектуру веб-додатку, включаючи структуру бази даних та API; реалізувати клієнтську частину з використанням HTML, CSS та JavaScript; розробити серверну частину на базі Node.js та Express.js для взаємодії з базою даних MySQL; інтегрувати механізми кошика, оформлення замовлень та зворотного зв'язку; реалізувати адміністративну панель для управління продуктами, замовленнями та повідомленнями; а також провести тестування та налагодження розробленої системи.

Розроблене програмне рішення має значну практичну цінність, оскільки пропонує готовий до впровадження інструмент для онлайн-продажу спортивних

добавок. Ця система дозволить оптимізувати бізнес-процеси, автоматизувати управління замовленнями та продуктами, покращити взаємодію з клієнтами та розширити ринкову присутність. Завдяки модульній архітектурі та використанню сучасних веб-технологій, веб-магазин є масштабованим і легко адаптується для інших товарних категорій, забезпечуючи гнучкість та ефективність у розвитку електронної комерції.

Кваліфікаційна робота бакалавра складається зі вступу, чотирьох основних розділів, висновків та переліку використаних джерел. У першому розділі проводиться аналіз предметної області та формулюються вимоги до системи. Другий розділ присвячений проектуванню архітектури, бази даних та API. У третьому розділі детально описується практична реалізація клієнтської та серверної частин веб-магазину, а також процес тестування.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз ринку спортивних добавок та існуючих рішень

Ринок спортивних добавок в Україні та світі демонструє стабільне зростання, що обумовлено підвищенням обізнаності населення щодо здорового способу життя, зростанням популярності фітнесу та спорту, а також прагненням до оптимізації фізичної форми та досягнень. Споживачі все частіше звертаються до біологічно активних добавок, таких як протеїни, амінокислоти, креатин, вітамінно-мінеральні комплекси та передтренувальні суміші, для підтримки свого організму, прискорення відновлення та покращення спортивних результатів. Цей попит створює сприятливі умови для розвитку онлайн-торгівлі, оскільки веб-магазини можуть запропонувати ширший асортимент, конкурентні ціни та зручність покупки порівняно з традиційними роздрібними точками.

1.1.1 Огляд сучасного ринку онлайн-продажів

Сучасний ринок онлайн-продажів спортивного харчування характеризується високою динамікою та конкуренцією. Зростання проникнення інтернету та розвиток мобільних технологій сприяли переходу значної частини комерції в онлайн-середовище. Для споживачів це означає доступ до широкого вибору товарів з різних куточків світу, можливість порівняння цін та характеристик, а також зручність доставки без необхідності відвідування фізичних магазинів. З боку продавців, онлайн-платформи дозволяють значно розширити географію продажів, зменшити операційні витрати, пов'язані з орендою та утриманням торгових площ, та ефективніше взаємодіяти з цільовою аудиторією через цифрові канали маркетингу. Особливістю ринку спортивних добавок є висока потреба у достовірній інформації про продукти, їх склад, застосування та потенційні ефекти, що робить акцент на якісному контенті та консультаційній підтримці.

1.1.2 Аналіз функціоналу конкурентних веб-магазинів

Аналіз функціоналу провідних гравців на ринку спортивних добавок виявив низку спільних та відмінних рис. Більшість успішних веб-магазинів пропонують інтуїтивно зрозумілий інтерфейс, ефективний пошук та розширені фільтри за категоріями (протеїн, креатин, BCAA тощо), виробниками, ціною, вагою та іншими параметрами. Важливою є наявність детальних сторінок продуктів з якісними зображеннями, повним описом, рекомендаціями щодо застосування та відгуками користувачів. Функціонал кошика зазвичай дозволяє додавати, видаляти та змінювати кількість товарів, а процес оформлення замовлення є максимально спрощеним, з можливістю вибору різних методів оплати та доставки (кур'єрська служба, поштові відділення). Деякі магазини також інтегрують системи лояльності, персоналізовані рекомендації та блоги з корисною інформацією.

Однак, були виявлені й недоліки, такі як застарілий дизайн, відсутність повної адаптивності під мобільні пристрої, складність навігації для нових користувачів, обмежені можливості для адміністративного управління замовленнями та відсутність зручного інструментарію для обробки звернень через форму зворотного зв'язку.

1.1.3 Визначення основних вимог до функціоналу веб-магазину

На основі проведеного аналізу ринку та функціоналу конкурентів, було визначено наступні основні вимоги до функціоналу розроблюваного веб-магазину спортивних добавок. Система повинна забезпечувати повноцінну реєстрацію та авторизацію користувачів, включаючи можливість створення облікового запису та входу для персоналізації досвіду, відстеження замовлень та доступу до спеціальних пропозицій. Важливим аспектом є поділ ролей на "користувач" та "адміністратор" для розмежування прав доступу.

Для зручності користувачів необхідний функціональний каталог продуктів, що дозволяє зручно переглядати товари, здійснювати пошук за

назвою, застосовувати фільтри за типом та виробником, а також сортувати за ціною та назвою. Кожен продукт повинен мати окрему детальну сторінку, яка містить повну інформацію: назву, опис, ціну, тип, виробника та зображення.

Ключовим елементом електронної комерції є кошик, який повинен надавати функціонал для додавання товарів, зміни їх кількості та видалення. Стан кошика буде зберігатися на стороні клієнта, наприклад, за допомогою cookie. Процес оформлення замовлення має бути інтуїтивно зрозумілим, дозволяючи користувачам вводити контактні дані, обирати зручний метод оплати та гнучкі опції доставки, включаючи доставку додому або у відділення Нової Пошти. Зареєстровані користувачі також повинні мати можливість переглядати історію своїх замовлень, відстежуючи їхній статус та деталі.

Для ефективної комунікації з клієнтами передбачається форма зворотного зв'язку, через яку користувачі зможуть надсилати повідомлення адміністрації магазину. Нарешті, для управління всіма аспектами магазину буде розроблена адміністративна панель. Вона надасть адміністраторам інструментарій для керування продуктами (додавання, редагування, видалення), перегляду та зміни статусів замовлень, а також для перегляду та управління повідомленнями зворотного зв'язку, включаючи зміну їхнього статусу. Ці вимоги ляжуть в основу проектування та реалізації веб-магазину, забезпечуючи його функціональність та відповідність потребам користувачів та адміністраторів.

1.2 Постановка завдання та вимоги до системи

Постановка завдання для розробки веб-магазину спортивних добавок полягає у створенні повноцінної, інтерактивної та безпечної онлайн-платформи, яка задовольнятиме потреби як кінцевих користувачів, так і адміністраторів. Система має забезпечувати ефективний процес купівлі-продажу, зручну навігацію по каталогу товарів, персоналізований досвід для зареєстрованих користувачів, а також надійний механізм обробки замовлень та зворотного зв'язку. Важливим аспектом є також забезпечення високого рівня безпеки даних та адаптивності інтерфейсу під різні пристрої.

1.2.1 Функціональні вимоги

Функціональні вимоги до веб-магазину визначають конкретні можливості, які повинна надавати система користувачам та адміністраторам. Система повинна забезпечувати повне управління користувачами, дозволяючи новим користувачам реєструватися, а існуючим – авторизуватися за допомогою електронної пошти та пароля. Також передбачається механізм виходу з облікового запису. Для адміністраторів має бути можливість розмежування прав доступу, що дозволяє їм виконувати операції управління контентом та замовленнями.

Для зручності користувачів необхідний функціональний каталог товарів. Користувачі повинні мати можливість переглядати повний каталог спортивних добавок, здійснювати пошук за назвою, фільтрувати товари за типом продукту (наприклад, протеїн, креатин) та виробником, а також сортувати їх за ціною та назвою. Кожен продукт повинен мати окрему сторінку з детальним описом, зображеннями, ціною та іншими характеристиками.

Система повинна надавати інтуїтивно зрозумілий кошик, де користувачі можуть додавати, видаляти та змінювати кількість обраних товарів. Процес оформлення замовлення має бути оптимізований, включаючи введення контактних даних, вибір зручного методу оплати (онлайн-картою, готівкою при отриманні, банківським переказом) та гнучкі опції доставки. Опції доставки включають можливість вибору між доставкою додому (з введенням повної адреси) та доставкою у відділення Нової Пошти (з введенням номера відділення).

Зареєстровані користувачі повинні мати доступ до особистого кабінету, де вони можуть переглядати історію своїх замовлень, включаючи статус кожного замовлення, дату, загальну суму та деталі замовлених товарів. Для ефективної комунікації з клієнтами передбачається наявність форми зворотного зв'язку, через яку користувачі зможуть надсилати запитання, пропозиції або скарги адміністрації магазину.

Нарешті, для адміністраторів передбачається окремий інтерфейс – адміністративна панель, яка надає повний контроль над вмістом магазину та

процесами замовлень. Це включає можливість додавання, редагування та видалення продуктів, управління статусами замовлень (наприклад, "В обробці", "Відправлено", "Виконано"), а також перегляд та зміна статусів повідомлень, отриманих через форму зворотного зв'язку.

1.2.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи та обмеження, яким вона повинна відповідати. Для веб-магазину спортивних добавок це:

1. Продуктивність: Система повинна забезпечувати швидке завантаження сторінок та оперативну обробку запитів, навіть при значному навантаженні. Час відгуку на запити користувачів не повинен перевищувати 2-3 секунди.

2. Надійність: Система має бути стабільною та доступною 24/7, за винятком планових технічних робіт. Передбачається механізм обробки помилок та резервного копіювання даних для запобігання втраті інформації.

3. Безпека: Забезпечення конфіденційності та цілісності даних користувачів є пріоритетом. Це включає захист від несанкціонованого доступу, використання хешування паролів, захист від SQL-ін'єкцій та XSS-атак, а також безпечну передачу даних (використання HTTPS у продакшн-середовищі).

4. Зручність використання (Usability): Інтерфейс повинен бути інтуїтивно зрозумілим та легким у використанні для всіх категорій користувачів, незалежно від їхнього рівня технічної підготовки. Навігація має бути логічною та послідовною.

5. Адаптивність (Responsiveness): Веб-магазин повинен коректно відображатися та функціонувати на різних пристроях (десктопи, планшети, смартфони) та з різними розмірами екранів, забезпечуючи оптимальний користувацький досвід.

6. Масштабованість: Архітектура системи повинна дозволяти її розширення та додавання нового функціоналу в майбутньому без суттєвих переробок.

7. Сумісність: Веб-магазин повинен коректно працювати у сучасних веб-браузерах (Chrome, Firefox, Safari, Edge).

1.3 Вибір технологій для реалізації

Вибір технологічного стеку є критично важливим етапом у розробці будь-якого веб-додатку, оскільки він визначає можливості, продуктивність, масштабованість та зручність подальшої підтримки системи. Для реалізації веб-магазину спортивних добавок було обрано наступний набір технологій, який забезпечує баланс між швидкістю розробки, гнучкістю та надійністю.

На клієнтській стороні (frontend) будуть використані стандартні веб-технології: HTML (HyperText Markup Language) для структурування контенту сторінок, CSS (Cascading Style Sheets) для стилізації та візуального оформлення інтерфейсу, а також JavaScript для забезпечення інтерактивності, динамічного завантаження даних та обробки подій користувача. Використання чистого JavaScript без важких фреймворків дозволить досягти високої продуктивності та мінімального розміру клієнтського коду, що позитивно вплине на швидкість завантаження сторінок. Для зберігання стану кошика на клієнтській стороні будуть використовуватися файли Cookie.

На серверній стороні (backend) обрано платформу Node.js у поєднанні з фреймворком Express.js. Node.js є кросплатформним середовищем виконання JavaScript, яке дозволяє створювати швидкі та масштабовані мережеві додатки. Express.js, як мінімалістичний та гнучкий веб-фреймворк для Node.js, спрощує розробку RESTful API, маршрутизації та обробки HTTP-запитів. Для управління сесіями користувачів буде використовуватися бібліотека express-session, що забезпечить безпечне зберігання інформації про авторизацію.

В якості системи управління базами даних (СУБД) обрано MySQL. Це одна з найпопулярніших реляційних баз даних, яка відрізняється високою продуктивністю, надійністю та широкою підтримкою. MySQL ідеально підходить для зберігання структурованих даних, таких як інформація про користувачів, продукти, замовлення та повідомлення зворотного зв'язку.

Взаємодія з базою даних буде здійснюватися за допомогою офіційного драйвера mysql для Node.js, який дозволяє ефективно виконувати SQL-запити та керувати пулом підключень для оптимізації продуктивності.

Таким чином, обраний технологічний стек (HTML, CSS, JavaScript на фронтенді, Node.js з Express.js на бекенді та MySQL як база даних) є оптимальним для створення сучасного, функціонального та продуктивного веб-магазину спортивних добавок, забезпечуючи гнучкість у розробці та подальшій підтримці системи.

1.4 Висновок до першого розділу

У першому розділі кваліфікаційної роботи було проведено комплексний аналіз предметної області, що включав огляд сучасного ринку онлайн-продажів спортивних добавок та детальний аналіз функціоналу існуючих конкурентних веб-магазинів. Цей аналіз дозволив виявити ключові тенденції ринку, визначити сильні сторони успішних рішень та ідентифікувати типові недоліки, які необхідно врахувати при розробці власної системи. На основі отриманих даних було сформульовано чітку постановку завдання та деталізовано функціональні та нефункціональні вимоги до майбутнього веб-магазину. Визначені вимоги охоплюють усі аспекти взаємодії користувача з системою, від реєстрації та перегляду каталогу до оформлення замовлення та зворотного зв'язку, а також забезпечують необхідний інструментарій для адміністративного управління.

Крім того, було обґрунтовано вибір технологічного стеку, що включає HTML, CSS, JavaScript для клієнтської частини, Node.js та Express.js для серверної частини, а також MySQL як систему управління базами даних. Цей вибір забезпечує надійну основу для подальшого проектування та реалізації веб-магазину, що відповідатиме сучасним стандартам якості та ефективності.

РОЗДІЛ 2. ПРОЕКТНА ЧАСТИНА

2.1 Архітектура веб-додатку

Ефективне проектування архітектури веб-додатку є фундаментальним етапом, що визначає його стабільність, продуктивність, масштабованість та легкість у підтримці. Для веб-магазину спортивних добавок було обрано класичну клієнт-серверну архітектуру, яка є найбільш поширеною та перевіреною моделлю для сучасних веб-додатків. Ця архітектура дозволяє чітко розділити відповідальність між компонентами, забезпечуючи гнучкість у розробці та розгортанні.

2.1.1 Опис клієнт-серверної архітектури

Клієнт-серверна архітектура передбачає розподіл обчислювальних завдань між двома основними компонентами: клієнтом та сервером. У контексті веб-додатку, клієнтом виступає веб-браузер користувача (наприклад, Google Chrome, Mozilla Firefox), який відповідає за відображення користувацького інтерфейсу, обробку взаємодії з користувачем та відправку запитів до сервера. Клієнтська частина реалізується за допомогою HTML, CSS та JavaScript. Сервер є центральним компонентом, який обробляє запити від клієнтів, виконує бізнес-логіку, взаємодіє з базою даних та надсилає відповіді назад клієнту. Серверна частина розроблена на базі Node.js з використанням фреймворку Express.js.
Взаємодія між клієнтом та сервером відбувається за протоколом HTTP/HTTPS. Коли користувач виконує певну дію у веб-браузері (наприклад, натискає кнопку "Додати до кошика" або заповнює форму замовлення), клієнт формує HTTP-запит (GET, POST, PUT, DELETE) і відправляє його на сервер. Сервер приймає цей запит, обробляє його відповідно до визначеної бізнес-логіки (наприклад, додає товар до бази даних, оновлює статус замовлення), і, за необхідності, взаємодіє з базою даних для отримання або збереження інформації. Після обробки запиту сервер формує HTTP-відповідь, яка може містити дані

(наприклад, список продуктів у форматі JSON), статус операції або повідомлення про помилку, і відправляє її назад клієнту. Клієнт, отримавши відповідь, оновлює свій інтерфейс відповідно до отриманих даних.

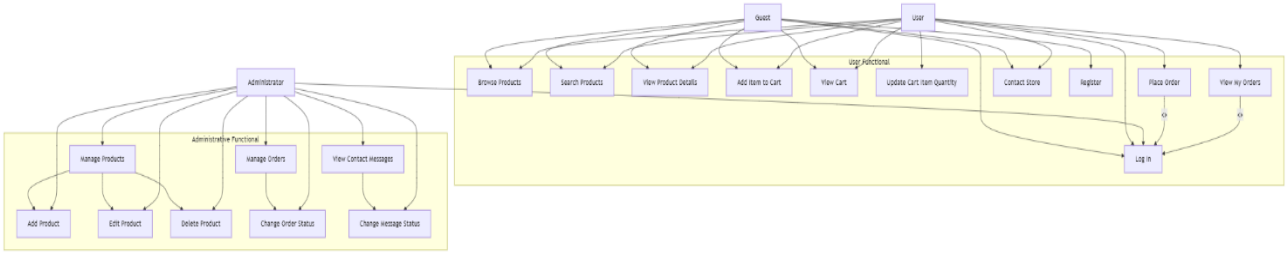

Рисунок 2.1 – Діаграма варіантів використання

Також на рисунку 2.1 зображена діаграма варіантів використання.

2.1.2 Схема взаємодії компонентів (frontend, backend, database)

Схема взаємодії компонентів у веб-магазині спортивних добавок є типовою для трирівневої архітектури: клієнтська частина (frontend), серверна частина (backend) та база даних (database).

1. Клієнтська частина (Frontend):

- Відповідає за відображення користувацького інтерфейсу, який складається з HTML-сторінок, стилізованих за допомогою CSS.
- JavaScript забезпечує інтерактивність: обробку подій (кліки, введення даних), динамічне оновлення контенту (наприклад, відображення товарів у кошику без перезавантаження сторінки) та формування AJAX-запитів до серверної частини.

Приклад формування запиту на отримання продуктів у script.js продемонстровано у лістингу 2.1.

Лістинг 2.1 – Фрагмент коду з public/js/script.js - відправка запиту на отримання продуктів

```
async function fetchProducts(filters = {}) {
const queryString = new URLSearchParams(filters).toString();
const response = await fetch(`/api/products?${queryString}`);
```

```
const products = await response.json();
// ... обробка та відображення продуктів }
```

2. Серверна частина (Backend):

- Розроблена на Node.js з використанням Express.js.
- Отримує HTTP-запити від клієнтської частини.
- Містить бізнес-логіку: аутентифікація користувачів, обробка замовлень, управління продуктами, обробка повідомлень зворотного зв'язку.
- Взаємодіє з базою даних MySQL для зберігання та отримання даних.
- Формує HTTP-відповіді та надсилає їх назад клієнту.

Приклад обробки запиту на сервері у server.js продемонстровано у лістингу 2.2.

Лістинг 2.2 – Фрагмент коду з server.js - обробка запиту на отримання продуктів

```
app.get('/api/products', (req, res) => {
     // ... логіка фільтрації та сортування ...
     db.getConnection((err, connection) => {
          if (err) { /* обробка помилки */ }
          connection.query(sql, params, (queryErr, results) => {
               connection.release(); // Повернення підключення в пул
               if (queryErr) { /* обробка помилки */ }
               res.json(results);
          });
     });
});
```

3. База даних (Database):

- Використовується MySQL для зберігання всіх даних веб-магазину: інформації про користувачів (users), продукти (products), замовлення (orders), деталі замовлень (order_items) та повідомлення зворотного зв'язку (contacts).
- Серверна частина надсилає SQL-запити до бази даних для виконання операцій CRUD (Create, Read, Update, Delete).

Приклад використання пулу підключень у db.js зображено у лістингу 2.3.

Лістинг 2.3 – Фрагмент коду з db.js - ініціалізація пулу підключень

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

```
const pool = mysql.createPool({
     host: process.env.DB_HOST,
     user: process.env.DB_USER,
     password: process.env.DB_PASSWORD,
     database: process.env.DB_NAME,
     connectionLimit: 10
});
module.exports = pool;
```

Ця архітектура забезпечує чітке розділення відповідальності, що спрощує розробку, тестування та подальшу підтримку кожного компонента окремо. Взаємодія між ними відбувається через стандартизовані протоколи та формати даних (HTTP, JSON), що робить систему гнучкою та дозволяє легко замінювати або оновлювати окремі частини без впливу на інші.

2.2 Проектування бази даних

Ефективне проектування бази даних є ключовим для забезпечення надійності, цілісності та продуктивності веб-магазину. База даних повинна бути спроектована таким чином, щоб ефективно зберігати інформацію про користувачів, продукти, замовлення та звернення, а також забезпечувати швидкий доступ до цих даних. Для веб-магазину спортивних добавок було обрано реляційну модель даних, що дозволяє чітко визначити сутності та зв'язки між ними, мінімізуючи надмірність та забезпечуючи цілісність даних.

2.2.1 Схема сутність-зв'язок (ER-діаграма)

Схема сутність-зв'язок (Entity-Relationship Diagram, ERD) є графічним представленням структури бази даних, що відображає сутності (об'єкти, які зберігають інформацію) та зв'язки між ними. Для веб-магазину спортивних добавок були визначені наступні основні сутності: "Користувачі" (users), "Продукти" (products), "Замовлення" (orders), "Елементи замовлення" (order_items) та "Контакти" (contacts). Користувачі представляють зареєстрованих відвідувачів магазину, продукти – асортимент товарів,

замовлення – інформацію про здійснені покупки, елементи замовлення – деталі про те, які саме продукти входять до кожного замовлення, а контакти – повідомлення, надіслані через форму зворотного зв'язку. ER діаграма є ключовим інструментом для візуалізації взаємозв'язків між даними, що забезпечує логічну організацію інформації та спрощує розробку запитів. Чітке визначення сутностей та їхніх атрибутів, а також встановлення типів зв'язків, є основою для побудови надійної та масштабованої бази даних. Спроєктовану ER-діаграму можна побачити на рисунку 2.2.

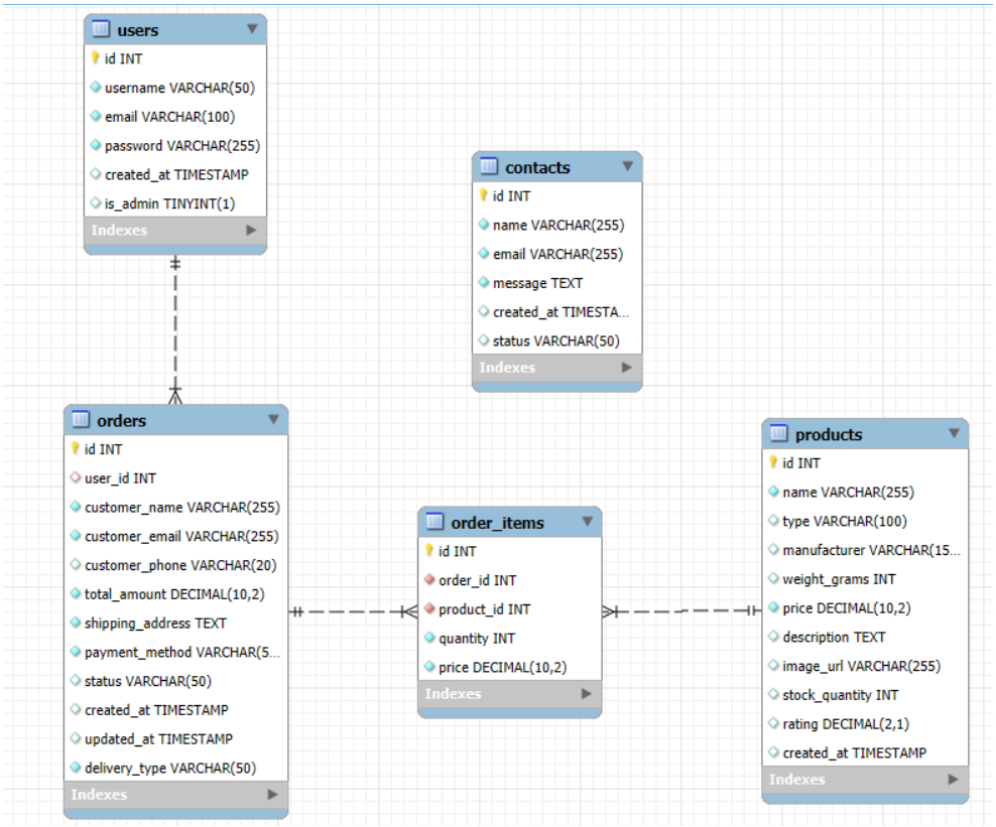

Рисунок 2.2 – Спроєктована ER-діаграма

Між цими сутностями встановлені логічні зв'язки. Кожен зареєстрований користувач може мати багато замовлень, але кожне замовлення, якщо воно прив'язане до облікового запису, належить лише одному користувачеві. Одне замовлення може містити багато різних продуктів, і ці деталі зберігаються в таблиці "Елементи замовлення". Кожен елемент замовлення, у свою чергу, посилається на конкретний продукт з каталогу. Таким чином, реалізується

зв'язок "багато-до-багатьох" між продуктами та замовленнями через проміжну таблицю "Елементи замовлення".

2.2.2 Опис таблиць бази даних

На основі визначених сутностей були спроектовані наступні таблиці бази даних MySQL. Кожна таблиця має чітко визначені поля та типи даних, що забезпечує структуроване зберігання інформації та ефективну взаємодію з даними. Нижче у лістингу 2.4 представлено SQL-сценарій для створення таблиці products.

Лістинг 2.4 – SQL-сценарій для створення таблиць бази даних

```
-- Таблиця `products` (продукти)
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL,
    type VARCHAR(100),
    manufacturer VARCHAR(100),
    weight_grams INT NOT NULL,
    image_url VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
);
```

Створені таблиці в MySQL зображені на рисунку 2.3.

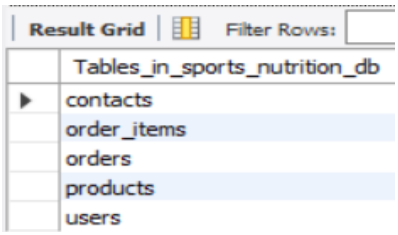

Рисунок 2.3 – Створені таблиці в MySQL

Кожна з цих таблиць відіграє ключову роль у функціонуванні веб-магазину. Таблиця users зберігає облікові дані користувачів, включаючи їхні імена, електронні пошти, хешовані паролі та роль адміністратора. products

містить всю необхідну інформацію про доступні для продажу спортивні добавки, такі як назва, опис, ціна, тип, виробник та посилання на зображення. orders фіксує загальні дані про кожне замовлення, включаючи інформацію про клієнта, загальну суму, адресу та деталі доставки, метод оплати та поточний статус. Для деталізації складу кожного замовлення використовується таблиця order_items, яка зв'язує замовлення з конкретними продуктами та фіксує кількість і ціну продукту на момент покупки. Нарешті, таблиця contacts призначена для зберігання повідомлень, що надходять від користувачів через форму зворотного зв'язку, дозволяючи відстежувати їхній статус обробки.

2.2.3 Обґрунтування структури таблиць та зв'язків

Структура таблиць та визначені зв'язки були розроблені відповідно до принципів нормалізації бази даних, що є фундаментальним для забезпечення цілісності, консистентності та ефективності зберігання даних. Основною метою нормалізації є мінімізація надмірності даних та уникнення аномалій при вставці, оновленні та видаленні інформації.

Використання окремих таблиць для users, products, orders, order_items та contacts дозволяє уникнути дублювання інформації. Наприклад, дані про продукт зберігаються лише один раз у таблиці products, а в order_items посилаються на нього за допомогою зовнішнього ключа product_id. Це забезпечує легкість оновлення інформації про продукт: достатньо змінити запис в одній таблиці, і ці зміни будуть відображені скрізь, де використовується цей продукт.

Зв'язки між таблицями реалізовані за допомогою зовнішніх ключів (FOREIGN KEY), що підтримує реферaційну цілісність. Наприклад, user_id в таблиці orders посилається на id в таблиці users з опцією ON DELETE SET NULL, що дозволяє зберігати історію замовлень навіть у випадку видалення облікового запису користувача. Для зв'язку між orders та order_items обрано ON DELETE CASCADE, що гарантує автоматичне видалення всіх елементів замовлення при видаленні самого замовлення, підтримуючи логічну цілісність даних. Натомість, для зв'язку order_items з products використано ON DELETE RESTRICT, що

запобігає видаленню продукту, якщо він вже є частиною оформленого замовлення, що захищає історичні дані.

Включення поля price безпосередньо в таблицю order_items є свідомим рішенням, що відходить від строгої третьої нормальної форми, але є загальноприйнятою практикою в системах електронної комерції. Це дозволяє фіксувати ціну продукту на момент замовлення, що є критично важливим, оскільки ціни в каталозі можуть змінюватися з часом. Без цього, історичні замовлення відображали б поточну ціну продукту, а не ту, за якою він був фактично проданий. Така денормалізація є виправданою для забезпечення точності фінансових даних та спрощення звітів.

Загалом, спроектована структура бази даних забезпечує необхідну гнучкість, надійність та ефективність для всіх операцій веб-магазину, від управління користувачами до обробки складних замовлень та звернень.

2.3 Проектування API

Проектування програмного інтерфейсу застосунків (API) є центральним елементом для забезпечення ефективної взаємодії між клієнтською та серверною частинами веб-додатку. Для веб-магазину спортивних добавок було обрано архітектурний стиль REST (Representational State Transfer), який є найбільш поширеним стандартом для розробки веб-сервісів. RESTful API забезпечує гнучкість, масштабованість та легкість у розумінні завдяки використанню стандартних HTTP-методів (GET, POST, PUT, DELETE) для виконання операцій над ресурсами. Це дозволяє клієнтській частині взаємодіяти з сервером через чітко визначені ендпоінти, обмінюючись даними у форматі JSON.

2.3.1 Опис RESTful API ендпоінтів

Розроблений API включає набір ендпоінтів, кожен з яких відповідає за певну функціональність та оперує конкретними ресурсами системи. Для управління користувачами та аутентифікації передбачені ендпоінти /api/register

(POST) для реєстрації нового користувача, /api/login (POST) для авторизації та створення сесії, /api/logout (POST) для завершення сесії, а також /api/check-auth (GET) для перевірки поточного статусу аутентифікації.

Управління продуктами реалізується через ендпоінт /api/products. Запит GET до нього дозволяє отримати список усіх продуктів, підтримуючи фільтрацію, пошук та сортування. Детальну інформацію про конкретний продукт можна отримати за допомогою GET-запиту до /api/products/:id, де :id — це ідентифікатор продукту. Для адміністраторів передбачені спеціальні ендпоінти: /api/admin/products (POST) для додавання нових продуктів, а також /api/admin/products/:id (PUT та DELETE) для редагування та видалення існуючих продуктів відповідно. Ці адміністративні операції вимагають відповідних прав доступу.

Функціонал управління кошиком та замовленнями включає ендпоінт /api/cart/items (POST) для отримання детальної інформації про товари в кошику на основі їхніх ID, які передаються з клієнтської сторони. Оформлення нового замовлення відбувається через /api/orders (POST). Зареєстровані користувачі можуть переглядати історію своїх замовлень, надсилаючи GET-запит до /api/my-orders, що вимагає аутентифікації. Адміністратори мають доступ до /api/admin/orders (GET) для перегляду всіх замовлень у системі та можуть оновлювати статус конкретного замовлення за допомогою PUT-запиту до /api/admin/orders/:id/status.

Для зворотного зв'язку передбачено ендпоінт /api/contact (POST) для надсилання повідомлень адміністрації. Адміністратори можуть переглядати всі отримані повідомлення через /api/admin/contacts (GET) та оновлювати їхній статус за допомогою PUT-запиту до /api/admin/contacts/:id/status.

2.3.2 Приклади запитів та відповідей

Для кращого розуміння взаємодії між клієнтською та серверною частинами, наведемо приклади типових HTTP-запитів та очікуваних відповідей у форматі JSON.

Приклад 1: Реєстрація нового користувача. Користувач надсилає POST-запит до /api/register з даними для реєстрації, такими як ім'я користувача, електронна пошта та пароль. У разі успіху сервер повертає статус 201 Created та повідомлення про успішну реєстрацію. Якщо користувач з такою електронною поштою вже існує, сервер відповість статусом 409 Conflict та відповідним повідомленням про помилку. Приклад наведений у лістингу 2.5.

Лістинг 2.5 – Приклад запиту реєстрації нового користувача

```
POST /api/register
Content-Type: application/json
{ "username": "testuser",
"email": "test@example.com",
"password": "password123" }
# Успішна відповідь
HTTP/1.1 201 Created
Content-Type: application/json
{
"message": "Реєстрація успішна! Тепер ви можете увійти."
}
# Відповідь з помилкою (email вже існує)
HTTP/1.1 409 Conflict
Content-Type: application/json
{
"error": "Користувач з таким email вже існує."
}
```

Реєстрація на сайті магазину зображена на рисунку 2.4 .

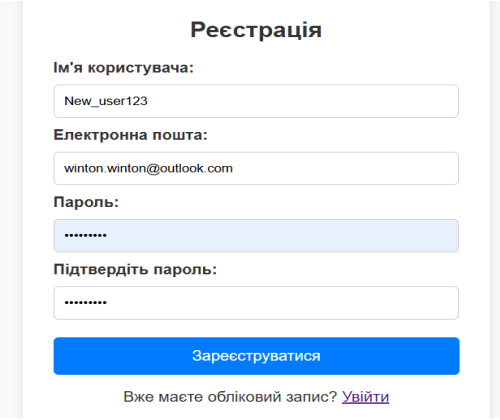

Рисунок 2.4 – Реєстрація на сайті магазину

Приклад 2: Отримання списку продуктів. Клієнт може надіслати GET-запит до /api/products, опціонально додаючи параметри фільтрації, пошуку та

сортування, наприклад, ?search=протеїн&type=Протеїн&sortBy=price_asc. Сервер поверне список об'єктів продуктів у форматі JSON зі статусом 200 OK. Кожен об'єкт продукту міститиме його ідентифікатор, назву, опис, ціну, тип, виробника, URL зображення та тимчасові мітки створення та оновлення. Приклад наведений у лістингу 2.6.

Лістинг 2.6 – Запит на отримання продуктів з фільтрацією

```
GET /api/products?search=протеїн&type=Протеїн&sortBy=price_asc
# Успішна відповідь
HTTP/1.1 200 OK
Content-Type: application/json
[
     {
          "id": 1,
          "name": "Сироватковий протеїн",
          "description": "Високоякісний протеїн...",
          "price": 850.00,
          "type": "Протеїн",
          "manufacturer": "Healthy Nutrition Co.",
          "image_url": "https://example.com/protein.jpg",
          "created_at": "2023-01-01T10:00:00.000Z",
          "updated_at": "2023-01-01T10:00:00.000Z"
          },
          {
          "id": 5,
          "name": "Протеїн для набору маси",
          "description": "Комплексний протеїн...",
          "price": 920.50, "type": "Протеїн",
          "manufacturer": "Premium Supplements",
          "image_url": "https://example.com/mass_protein.jpg",
          "created_at": "2023-01-05T12:00:00.000Z",
          "updated_at": "2023-01-05T12:00:00.000Z"
     }
```

Приклад 3: Оформлення замовлення. Для оформлення замовлення клієнт надсилає POST-запит до /api/orders, передаючи масив товарів у кошику (з їх ID та кількістю), контактні дані клієнта (ім'я, email), повну адресу доставки, обраний метод оплати, тип доставки та деталі доставки. У разі успішного оформлення сервер поверне статус 201 Created та повідомлення про успішне створення замовлення з його ідентифікатором. Якщо кошик порожній або не заповнені всі обов'язкові поля, сервер відповість статусом 400 Bad Request та відповідним

повідомленням про помилку. Приклад запиту на оформлення замовлення наведений у лістингу 2.7.

Лістинг 2.7 – Запит на оформлення замовлення

```
POST /api/orders
Content-Type: application/json

{
    "cartItems": [
        { "id": 1, "quantity": 2 },
        { "id": 3, "quantity": 1 }
    ],
    "customerName": "Іван Петров",
    "customerEmail": "ivan.petrov@example.com",
    "customerPhone": "+380879675647",
    "shippingAddress": "м. Київ, вул. Хрещатик, 10, кв. 5",
    "paymentMethod": "Карта онлайн",
    "deliveryType": "Додому",
}

# Успішна відповідь
HTTP/1.1 201 Created
Content-Type: application/json

{
    "message": "Замовлення успішно оформлено!",
    "orderId": 123
}

# Відповідь з помилкою (кошик порожній)
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
    "error": "Кошик порожній."
}
```

Приклад 4: Надсилання повідомлення зворотного зв'язку. Користувач надсилає POST-запит до /api/contact, вказуючи своє ім'я, електронну пошту та текст повідомлення. У разі успішного надсилання сервер поверне статус 201 Created та підтверджуюче повідомлення. Приклад наведений у лістингу 2.8.

Лістинг 2.8 – Запит на надсилання повідомлення

```
POST /api/contact
Content-Type: application/json
```

```
{
    "name": "Олена Ковальчук",
    "email": "olena.kovalchuk@example.com",
    "message": "Хотіла б дізнатися про наявність товару X."
}

# Успішна відповідь
HTTP/1.1 201 Created
Content-Type: application/json

{
    "message": "Ваше повідомлення успішно відправлено!"
}
```

Ці приклади демонструють, як клієнтська частина взаємодіє з серверною через API, надсилаючи запити та отримуючи відповіді, що дозволяє реалізувати весь необхідний функціонал веб-магазину.

2.4 Проектування користувацького інтерфейсу (UI/UX)

Проектування користувацького інтерфейсу (UI) та користувацького досвіду (UX) є невід'ємною частиною розробки веб-магазину, оскільки саме вони визначають, наскільки зручним, інтуїтивно зрозумілим та привабливим буде взаємодія користувача із системою. Метою UI/UX проектування було створення чистого, сучасного та функціонального інтерфейсу, який забезпечить легку навігацію, швидкий доступ до інформації та комфортне виконання ключових операцій, таких як пошук товарів, додавання їх до кошика та оформлення замовлення. Особлива увага приділялася візуальній привабливості та логічній послідовності елементів для забезпечення позитивного користувацького досвіду.

2.4.1 Макет основних сторінок

Макет веб-магазину спортивних добавок був розроблений з урахуванням типових патернів електронної комерції, забезпечуючи послідовність та впізнаваність елементів на різних сторінках. Кожна сторінка має спільний хедер, що включає назву магазину ("IronLab") та навігаційне меню з посиланнями на основні розділи: "Головна", "Продукти", "Про нас", "Контакти", "Кошик" (з

динамічним лічильником товарів), а також посилання "Увійти/Вийти" та приховані за замовчуванням посилання на "Мої Замовлення" та "Адмін-панель" для авторизованих користувачів. Футер містить стандартну інформацію.

Головна сторінка (index.html) служить вхідною точкою для користувачів. Вона містить секцію "Hero" з закликом до дії і "Популярні продукти", яка відображає підбірку 4 товарів з каталогу. На рисунку 2.5 зображено головну.

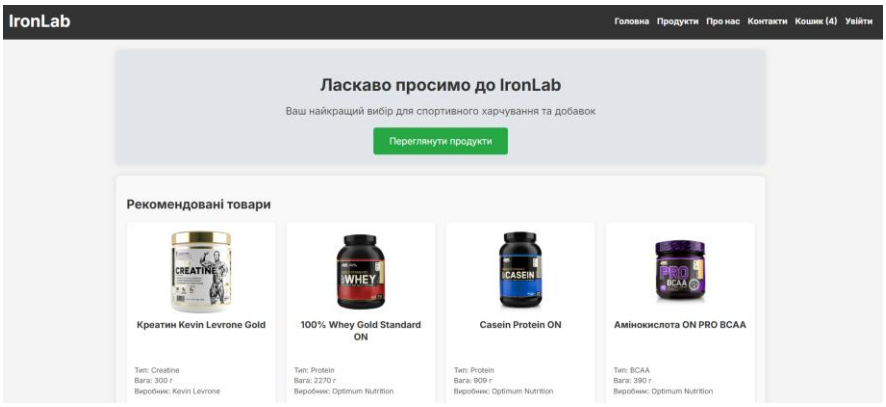

Рисунок 2.5 – Головна сторінка

Сторінка продуктів (products.html) представляє повний каталог доступних спортивних добавок. Тут блок пошуку, фільтрації, сортування. Продукти відображаються у вигляді карток, кожна з яких містить зображення, назву, короткий опис, ціну та кнопку "Додати до кошика". На рисунку 2.6 можна побачити як виглядає ця сторінка з фільтрацією та сортуванням за ціною.

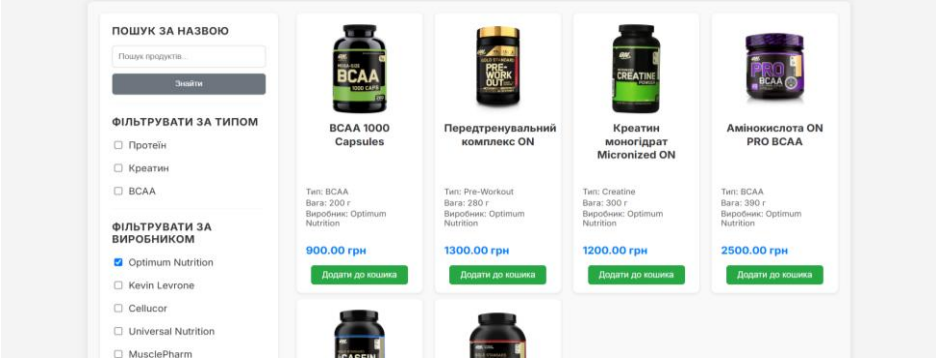

Рисунок 2.6 – Сторінка продуктів

Сторінка детального продукту (product.html) надає розгорнуту інформацію про обраний товар. Вона включає велике зображення продукту, повний опис,

актуальну ціну, тип, виробника та кнопку для додавання товару до кошика. Ця сторінка зображена на рисунку 2.7.

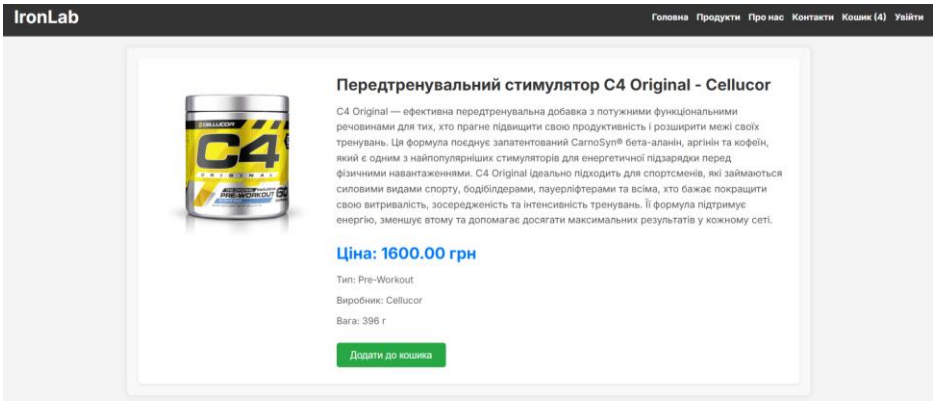

Рисунок 2.7 – Сторінка продукту

Сторінка кошика (cart.html) відображає перелік усіх доданих товарів, їхню кількість, ціну за одиницю та загальну вартість кожного товару, а також загальну суму замовлення. На цій сторінці реалізовано функціонал для зміни кількості товарів та їх видалення. Ключовим елементом є форма оформлення замовлення, яка включає поля для введення контактних даних клієнта (ім'я, email), вибір типу доставки (додому або у відділення Нової Пошти з відповідними полями для введення деталей), та вибір методу оплати. Кошик зображений на рисунку 2.8.

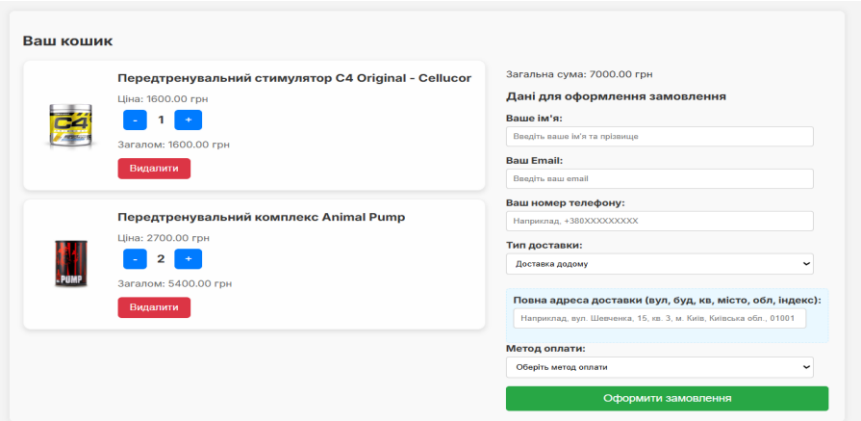

Рисунок 2.8 – Сторінка кошику

Сторінки реєстрації (register.html) та входу (login.html) містять прості та зрозумілі форми для створення нового облікового запису та авторизації відповідно.

Сторінка контактів (contact.html) надає інформацію про способи зв'язку з магазином та інтегровану форму зворотного зв'язку.

Сторінка "Мої замовлення" (my-orders.html) доступна лише для зареєстрованих користувачів і відображає список усіх їхніх попередніх замовлень з детальною інформацією про кожен. Сторінка мої замовлення зображена на рисунку 2.9.

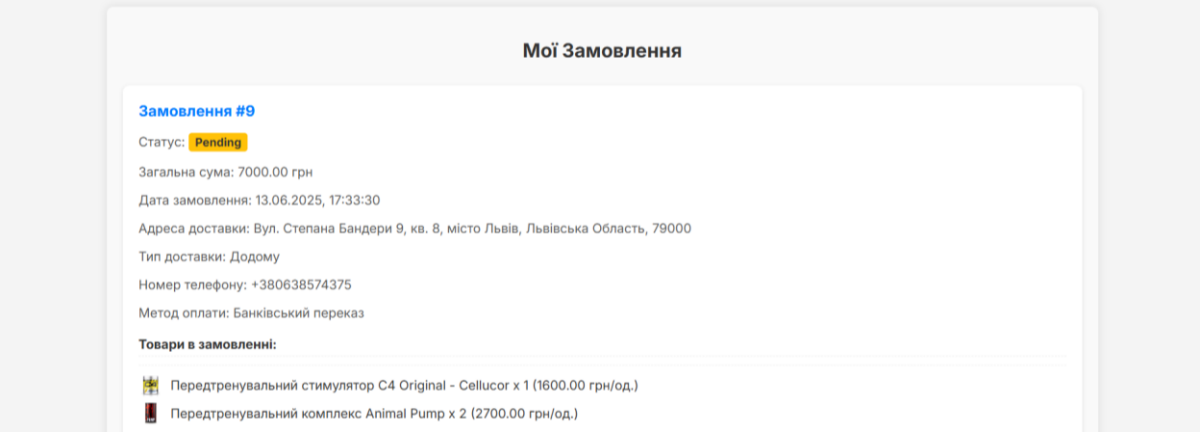

Рисунок 2.9 – Сторінка моїх замовлень

Адміністративна панель (admin.html) є центральним інструментом для управління магазином. Вона розділена на секції: "Управління Продуктами" (з формами для додавання/редагування та таблицею для перегляду/видалення продуктів), ця форма зображена на рисунку 2.10 .

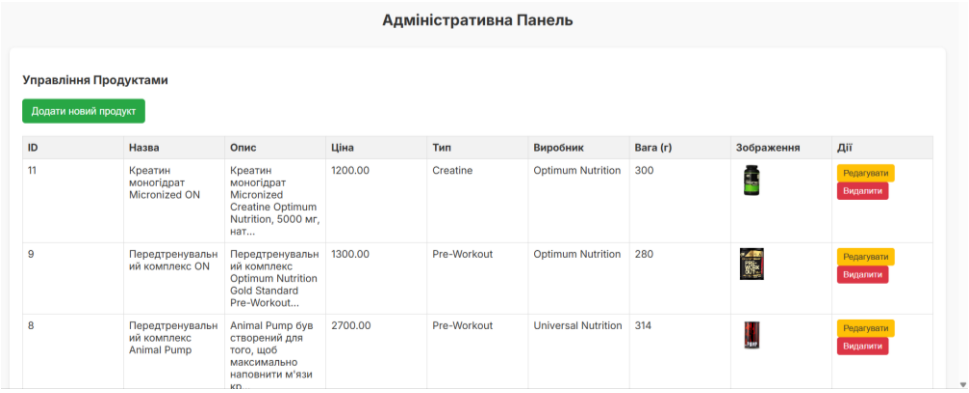

Рисунок 2.10 – Форма для редагування та додавання продуктів

Як можна побачити, що розроблена доволі зручна панель для адміністрування.

"Управління Замовленнями" (з таблицею всіх замовлень та можливістю зміни їх статусу), цю форму можна побачити на рисунку 2.11 .

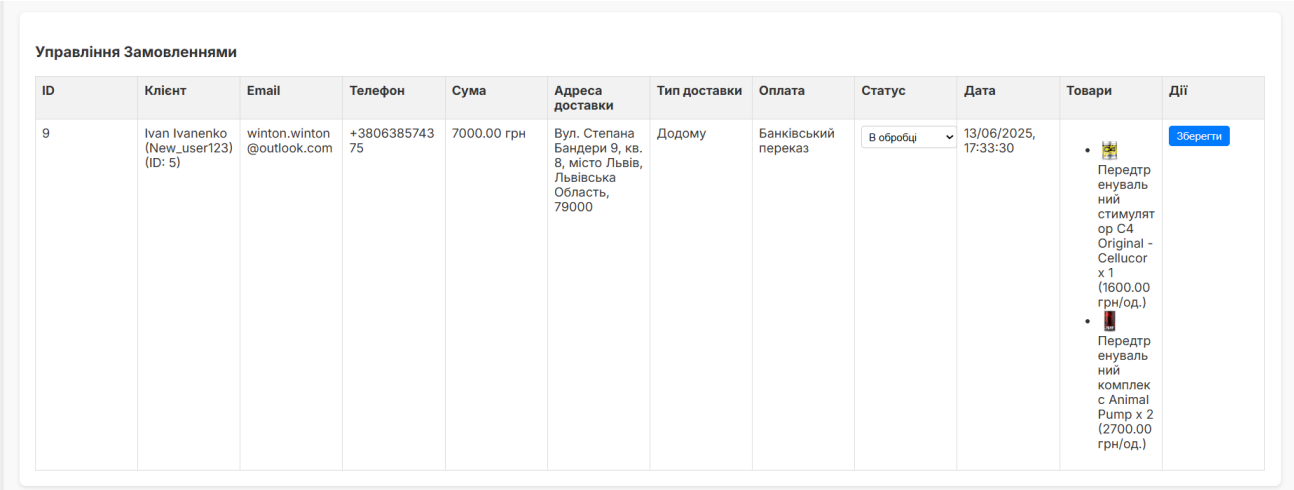

Рисунок 2.11 – Форма замовлень

"Повідомлення Зворотного Зв'язку" (з таблицею отриманих повідомлень та можливістю зміни їх статусу) зображена на рисунку 2.12 .

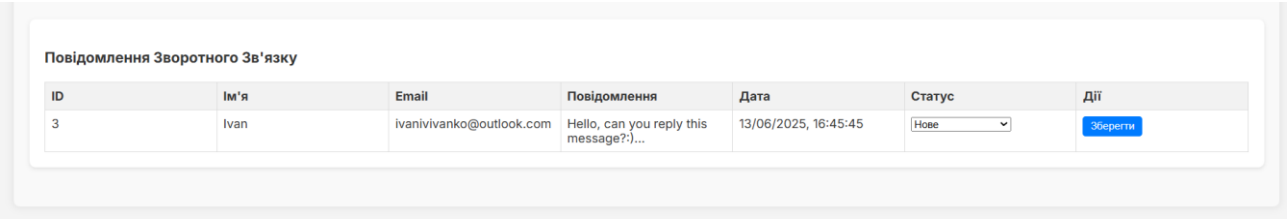

Рисунок 2.12 – Форма повідомлень зворотнього зв’язку

Як можна побачити за допомогою цих форм можна доволі зручно проводити адміністрування веб-магазину.

2.4.2 Принципи адаптивного дизайну

Для забезпечення оптимального користувацького досвіду на широкому спектрі пристроїв, від настільних комп'ютерів до смартфонів, веб-магазин був розроблений з використанням принципів адаптивного дизайну (Responsive Web Design). Це означає, що макет та елементи інтерфейсу автоматично адаптуються

до розміру екрану користувача, забезпечуючи зручність перегляду та взаємодії без необхідності горизонтальної прокрутки.

Адаптивність досягається за рахунок кількох ключових підходів. По-перше, використовуються гнучкі сітки та макети на основі Flexbox та Grid-систем CSS, які дозволяють елементам динамічно змінювати свій розмір та розташування. Наприклад, картки продуктів на сторінці каталогу автоматично перебудовуються, щоб займати оптимальну кількість колонок залежно від доступної ширини екрану. По-друге, зображення та медіа-контент оптимізовані для адаптивності за допомогою властивості max-width: 100%, що гарантує їхнє масштабування відповідно до розміру контейнера.

По-третє, активно застосовуються медіа-запити CSS (@media rules). Це дозволяє застосовувати різні стилі CSS залежно від характеристик пристрою, таких як ширина екрану. Наприклад, на мобільних пристроях навігаційне меню в хедері може змінювати своє розташування з горизонтального на вертикальне, а елементи форм та кнопок можуть розширюватися на всю ширину екрану для зручності натискання. Це було продемонстровано у попередніх змінах до файлу style.css для хедера.

Всі інтерактивні елементи, такі як кнопки та посилання, мають достатній розмір та відступи, щоб забезпечити зручне натискання на сенсорних екранах. Завдяки цим принципам, веб-магазин забезпечує послідовний та приємний користувацький досвід незалежно від пристрою, яким користується відвідувач.

2.5 Висновок до другого розділу

У другому розділі кваліфікаційної роботи було детально розглянуто та спроектовано ключові архітектурні компоненти веб-магазину спортивних добавок. Була обґрунтована та описана клієнт-серверна архітектура, що забезпечує чітке розділення відповідальності між фронтендом, бекендом та базою даних, а також схеми їхньої взаємодії. Особливу увагу було приділено проектуванню бази даних MySQL, де були визначені сутності, їхні зв'язки, а також детально описані структури таблиць users, products, orders, order_items та

contacts з відповідними SQL-сценаріями. Обґрунтування структури таблиць підкреслило принципи нормалізації та стратегії забезпечення цілісності даних, включаючи фіксацію ціни товару на момент замовлення.

Крім того, було спроектовано RESTful API, що включає повний набір ендпоінтів для управління всіма ресурсами системи, від аутентифікації користувачів до обробки замовлень та зворотного зв'язку, з наведенням прикладів запитів та відповідей. Нарешті, було розроблено принципи проектування користувацького інтерфейсу та досвіду, включаючи макети основних сторінок та реалізацію адаптивного дизайну, що забезпечує зручність використання на різних пристроях.

Результати проектування, представлені у цьому розділі, закладають міцну основу для практичної реалізації веб-магазину, що буде розглянута у наступному розділі.

РОЗДІЛ 3. ПРАКТИЧНА ЧАСТИНА

3.1 Реалізація клієнтської частини (Frontend)

Реалізація клієнтської частини веб-магазину є ключовим етапом, що забезпечує безпосередню взаємодію користувача із системою. Вона включає розробку структури сторінок за допомогою HTML, візуальне оформлення за допомогою CSS та створення інтерактивної логіки за допомогою JavaScript. Головною метою було створення інтуїтивно зрозумілого, привабливого та адаптивного інтерфейсу, що забезпечує зручний та ефективний користувацький досвід.

3.1.1 Розробка HTML-структури сторінок

HTML був використаний для створення семантичної та логічної структури всіх веб-сторінок магазину. Кожна сторінка розроблена з використанням сучасних елементів HTML5, що покращує доступність та оптимізацію для пошукових систем. Основні сторінки, такі як index.html (головна), products.html (каталог товарів), product.html (детальний опис продукту), cart.html (кошик), login.html (вхід), register.html (реєстрація), contact.html (контакти), my-orders.html (мої замовлення) та admin.html (адмін-панель), мають послідовну структуру.

Кожна сторінка містить спільний header з назвою магазину "IronLab" та навігаційним меню, а також footer з інформацією про авторські права. Основний контент розміщується всередині тегу main. Для динамічного відображення продуктів, елементів кошика та списків замовлень використовуються порожні контейнери (div або table), які заповнюються даними за допомогою JavaScript. Структура картки продукту на сторінці каталогу наведена у лістингу 3.1.

Лістинг 3.1 – Фрагмент коду з public/products.html - структура картки продукту

```
<div class="product-card">
```

```
    ><a href="/product.html?id=[ID_продукту]">
    <img src="[URL_зображення]" alt="[Назва продукту]">
    <h3 [Назва продукту]</h3>
    <p>[Короткий опис]...</p>
    <p class="price">[Ціна] грн</p>
    </a>
    <button          class="add-to-cart-btn"          data-product-
id="[ID_продукту]">Додати до кошика</button>
</div>
```

Аналогічно, форми для реєстрації, входу, оформлення замовлення та зворотного зв'язку побудовані з використанням семантичних тегів form, label, input, select та textarea, що забезпечує їхню доступність та правильну структуру для обробки JavaScript.

3.1.2 Створення CSS-стилів

CSS (Cascading Style Sheets) використовувався для візуального оформлення веб-магазину, забезпечуючи його сучасний та привабливий вигляд. Всі стилі централізовано зберігаються у файлі public/css/style.css. Основні принципи стилізації включали: забезпечення читабельності тексту, використання контрастних кольорів, створення інтуїтивно зрозумілих візуальних ієрархій та привабливих інтерактивних елементів.

Особлива увага приділялася реалізації адаптивного дизайну. Застосування гнучких макетів за допомогою Flexbox та CSS Grid дозволило елементам інтерфейсу динамічно адаптуватися до різних розмірів екранів. Медіа-запити (@media rules) були використані для застосування специфічних стилів на мобільних пристроях, забезпечуючи оптимальний користувацький досвід незалежно від пристрою. Такий підхід забезпечить хороше відображення та функціональність веб-магазину на смартфонах, планшетах та десктопах. Це значно покращить доступність і зручність використання для всіх категорій відвідувачів. Наприклад, хедер магазину, включаючи назву "IronLab" та навігаційне меню (Продукти, Кошик, Контакти), був стилізований для відображення в один рядок на великих екранах та переходу у стовпцевий режим

на менших екранах, фрагмент коду з цими стилями хедера наведений у лістингу 3.2.

Лістинг 3.2 – Фрагмент коду з public/css/style.css - стилі хедера

```
header {
    background-color: #333;
    color: white;
    padding: 1em 20px;
    display: flex; /* Використовуємо flexbox */
    justify-content: space-between;
    align-items: center;
    flex-wrap: wrap;
}

/* ... інші стилі хедера ... */

@media (max-width: 768px) {
    header {
        flex-direction: column; /* На малих екранах елементи хедера
розташовуються вертикально */
        text-align: center;
    }
    /* ... інші адаптивні стилі ... */
}
```

Крім того, було розроблено стилі для карток продуктів, форм, кнопок, елементів кошика, таблиць адміністративної панелі та карток замовлень, забезпечуючи єдиний візуальний стиль по всьому додатку.

3.1.3 Розробка JavaScript-логіки (управління станом, динамічний контент, логіка адміністрування)

JavaScript є основою інтерактивності та динамічності веб-магазину, реалізований у файлі public/js/script.js. Він відповідає за управління станом, динамічне завантаження та відображення контенту, а також за логіку адміністративної панелі.

Управління станом: Стан кошика користувача зберігається безпосередньо у браузерних файлах Cookie. Це дозволяє зберігати обрані товари між сесіями та перезавантаженнями сторінок без необхідності постійної взаємодії з сервером для кожного елемента кошика. Функції getCartFromCookie() та

saveCartToCookie() забезпечують читання та запис даних кошика. Кількість товарів у кошику динамічно оновлюється на всіх сторінках за допомогою функції updateCartCount().

Динамічний контент та взаємодія з користувачем: JavaScript відповідає за асинхронне завантаження даних з сервера та їх динамічне відображення на сторінках. Функція fetchProducts() отримує список продуктів з API та рендерить їх у вигляді карток на головній сторінці та сторінці каталогу, включаючи функціонал пошуку, фільтрації та сортування. Аналогічно, renderCartItems() завантажує деталі товарів з кошика та відображає їх на сторінці кошика, дозволяючи користувачам змінювати кількість товарів або видаляти їх. Обробники подій прив'язані до кнопок "Додати до кошика", "Збільшити/зменшити кількість" та "Видалити з кошика", код наведений у лістингу 3.3.

Лістинг 3.3 – Фрагмент коду з public/js/script.js - додавання товару до кошика

```
function addProductToCart(productId) {
    const cart = getCartFromCookie();
    cart[productId] = (cart[productId] || 0) + 1;
    saveCartToCookie(cart);
    alert('Товар додано до кошика!');
    updateCartCount();
}
```

Форми реєстрації, входу, оформлення замовлення та зворотного зв'язку обробляються за допомогою JavaScript, який збирає дані з полів, надсилає їх на сервер через AJAX-запити (використовуючи fetch API) та обробляє відповіді. Наприклад, форма оформлення замовлення динамічно змінює поля вводу залежно від обраного типу доставки (додому або Нова Пошта).

Логіка адміністрування: Для адміністративної панелі JavaScript забезпечує повний контроль над продуктами, замовленнями та повідомленнями зворотного зв'язку. Функція fetchAdminProducts() завантажує та відображає таблицю продуктів, а також прив'язує обробники подій до кнопок "Редагувати" та "Видалити". Функції editProduct() та saveProduct() керують відображенням та

відправкою даних форми додавання/редагування продукту, а deleteProduct() обробляє запити на видалення. Приклад коду з функцією saveProduct() наведена у лістингу 3.4.

Лістинг 3.4 – Фрагмент коду з public/js/script.js - обробник збереження продукту

```
async function saveProduct(event) {
    event.preventDefault();
    const productId = document.getElementById('product-id').value;
    // ... збір даних форми ...
    const productData = { name, description, price, type,
manufacturer, image_url };

    try {
        let response;
        if (productId) {
            response = await
fetch(`/api/admin/products/${productId}`, { method: 'PUT', /* ... */
body: JSON.stringify(productData) });
        } else {
            response = await fetch('/api/admin/products', { method:
'POST', /* ... */ body: JSON.stringify(productData) });
        }
        const result = await response.json();
        if (response.ok) {
            alert(result.message);
            document.getElementById('product-edit-
form').style.display = 'none';
            fetchAdminProducts(); // Оновлення списку
        } else { /* ... */ }
    } catch (error) { /* ... */ }
}
```

Отже, після збереження продукту він має додатися в нашу базу даних та поповнити наш список.

Аналогічно, fetchAdminOrders() та fetchAdminContacts() завантажують та відображають списки замовлень та повідомлень, дозволяючи адміністраторам змінювати їхні статуси через updateOrderStatus() та updateContactStatus(). Функція fetchUserOrders() забезпечує відображення історії замовлень для авторизованих користувачів. Загалом, JavaScript є сполучною ланкою між користувацьким інтерфейсом та серверним API, забезпечуючи повноцінну та динамічну роботу веб-магазину.

3.2 Реалізація серверної частини (Backend)

Серверна частина веб-магазину є фундаментом для обробки бізнес-логіки, взаємодії з базою даних та надання даних клієнтській частині через API. Реалізація бекенду здійснювалася з використанням Node.js та фреймворку Express.js, що забезпечує високу продуктивність та гнучкість.

3.2.1 Налаштування Node.js та Express.js

Розробка серверної частини розпочалася з налаштування середовища Node.js та ініціалізації проекту. Були встановлені необхідні залежності, такі як express для створення веб-сервера, mysql для взаємодії з базою даних, dotenv для управління змінними оточення, bcrypt для хешування паролів та express-session разом з cookie-parser для управління сесіями користувачів. Для встановлення цих залежностей було використано наступну команду: npm install express mysql dotenv bcrypt express-session cookie-parser.

Основний файл сервера, server.js, відповідає за конфігурацію Express.js, підключення до бази даних та визначення всіх API-маршрутів. Express.js був налаштований для обробки статичних файлів (HTML, CSS, JavaScript) з директорії public, а також для парсингу JSON-даних та URL-кодованих даних, що надходять у тілі HTTP-запитів. Конфігурація сесій забезпечує механізм для підтримки стану користувача між запитами, використовуючи куки для ідентифікації сесії, у лістингу 3.5 наведений фрагмент з коду.

Лістинг 3.5 – Фрагмент коду з server.js - налаштування Express.js та сесій

```
const express = require('express');
const db = require('./db'); // Тепер db - це пул підключень з db.js
const path = require('path');
const dotenv = require('dotenv');
const bcrypt = require('bcrypt');
const session = require('express-session');
const cookieParser = require('cookie-parser'); // Додано cookie-
parser
dotenv.config(); // Завантажуємо змінні оточення з .env файлу
const app = express();
```

```
const port = process.env.PORT || 3000;
// Middleware для обробки статичних файлів (HTML, CSS, JS)
app.use(express.static(path.join(__dirname, 'public')));
app.use(express.json()); // Дозволяє Express парсити JSON-дані з тіла запитів
app.use(express.urlencoded({ extended: true })); // Дозволяє Express парсити URL-кодовані дані
app.use(cookieParser()); // Використовуємо cookie-parser
app.use(session({ // Конфігурація сесій
    secret: process.env.SESSION_SECRET || 'supersecretkey', // Використовуйте змінну оточення
    resave: false,
    saveUninitialized: false,
    cookie: {
        secure: process.env.NODE_ENV === 'production', // true тільки для HTTPS в production
        httpOnly: true,
        maxAge: 24 * 60 * 60 * 1000 // 24 години
    }
}));
// Запуск сервера
app.listen(port, () => {
    console.log(`Сервер запущено на http://localhost:${port}`);
});
```

Після успішного запуску має з’явитися вікно, яке повідомить, що сервер запущено і на якому хості. Конфігурація сесій також дуже важливий елемент для підтримки стану користувача, використовуючи для цього кукі.

3.2.2 Взаємодія з базою даних MySQL

Взаємодія серверної частини з базою даних MySQL реалізована через пул підключень, що є оптимальним рішенням для веб-додатків. Пул підключень, ініціалізований у файлі db.js, дозволяє ефективно керувати з'єднаннями до бази даних, повторно використовуючи існуючі підключення замість створення нових для кожного запиту. Це значно покращує продуктивність та зменшує навантаження на сервер бази даних.

Файл db.js експортує об'єкт пулу, який потім імпортується у server.js. У всіх маршрутах, що вимагають взаємодії з базою даних, використовується метод db.getConnection() для отримання підключення з пулу. Після виконання запиту,

підключення обов'язково повертається в пул за допомогою connection.release(), що запобігає витокам ресурсів. Ініціалізація пулу підключень є у лістингу 3.6.

Лістинг 3.6 – Фрагмент коду з db.js - ініціалізація пулу підключень

```
const mysql = require('mysql');
const dotenv = require('dotenv');
dotenv.config(); // Завантажуємо змінні оточення
const pool = mysql.createPool({
    host: process.env.DB_HOST,
    user: process.env.DB_USER,
    password: process.env.DB_PASSWORD,
    database: process.env.DB_NAME,
    connectionLimit: 10, // Максимальна кількість одночасних підключень
    waitForConnections: true,
    queueLimit: 0
});
pool.getConnection((err, connection) => {
    if (err) {
        console.error('Помилка підключення до пулу бази даних:', err.message); // Помилка підключення
        process.exit(1);
    }
    console.log('Пул підключень до бази даних MySQL успішно ініціалізовано.'); // Пул ініціалізовано
    connection.release(); // Повернути підключення
});
module.exports = pool;
```

Ініціалізація пулу підключень є важливим кроком для створення інтернет магазину, атже необхідне правильне налаштування, інакше система не буде працювати.

3.2.3 Реалізація API-маршрутів

Усі API-маршрути, спроектовані у Розділі 2.3, були реалізовані у файлі server.js за допомогою Express.js. Кожен маршрут відповідає за обробку конкретного HTTP-запиту (GET, POST, PUT, DELETE) до певного ресурсу.

Наприклад, маршрут для реєстрації користувача приймає дані через POST-запит, хешує пароль за допомогою bcrypt та зберігає інформацію у таблиці users. Маршрути для продуктів дозволяють отримувати, додавати, редагувати та

видаляти дані, взаємодіючи з таблицею products. Особлива увага приділена маршрутам для замовлень, де реалізована логіка перевірки цін товарів, створення нового замовлення та додавання елементів замовлення в рамках однієї транзакції. Всі зміни успішні, або жодна з них не застосовується, що запобігає неконсистентності даних. У лістингу 3.7 наведений код з оформленням замовлення.

Лістинг 3.7 – Фрагмент коду з server.js - обробка оформлення замовлення

```
const orderSql = 'INSERT INTO orders (...) VALUES (?,...)';
const orderResult = await new Promise((resolve, reject) => {
    connection.query(orderSql, [...], (queryErr, result) => {
        if (queryErr) reject(queryErr);
        resolve(result);
    });
});
const orderId = orderResult.insertId;
const orderItemsSql = 'INSERT INTO order_items (order_id,
product_id, quantity, price) VALUES (?)';
const orderItemsValues = cartItems.map(item => [orderId, item.id,
item.quantity, productPrices[item.id]]);
await new Promise((resolve, reject) => {
    connection.query(orderItemsSql, [orderItemsValues], (queryErr,
result) => {
        if (queryErr) reject(queryErr);
        resolve(result);
    });
});
await new Promise((resolve, reject) => {
    connection.commit((commitErr) => { // Коміт транзакції
        if (commitErr) reject(commitErr);
        resolve();
    });
});
connection.release(); // Повернення підключення
res.status(201).json({ message: 'успішно', orderId: orderId });
} catch (transactionError) {
    connection.rollback(() => { // Відкат транзакції
        connection.release(); // Повернення підключення
        console.error('Помилка транзакції при оформленні
замовлення:', transactionError); // Помилка транзакції
        res.status(500).json({ error: 'Помилка сервера при
оформленні замовлення: ' + transactionError.message }); // Пом. Серв
```

Після успішного замовлення має з’явитися у нашій базі даних.

3.2.4 Забезпечення безпеки

Безпека є одним з найважливіших аспектів при розробці веб-додатку. У веб-магазині спортивних добавок було реалізовано кілька ключових механізмів безпеки:

- Хешування паролів: Паролі користувачів зберігаються у базі даних виключно у хешованому вигляді за допомогою бібліотеки bcrypt. Це запобігає прямому доступу до паролів навіть у випадку компрометації бази даних.
- Управління сесіями: Для аутентифікації користувачів використовується express-session, що створює безпечні сесії. Сесійні куки налаштовані з опціями secure: true (для HTTPS у продакшн) та httpOnly: true, що захищає їх від XSS-атак та несанкціонованого доступу з клієнтського JavaScript.
- Захист від SQL-ін'єкцій: Усі SQL-запити виконуються з використанням параметризованих запитів (prepared statements), де значення передаються окремо від SQL-коду. Це ефективно запобігає SQL-ін'єкціям, оскільки вхідні дані автоматично екрануються драйвером MySQL.
- Контроль доступу (Авторизація): Застосовані middleware-функції requireAuth та requireAdmin для захисту маршрутів, що вимагають аутентифікації або прав адміністратора. requireAuth перевіряє наявність активної сесії користувача, а requireAdmin додатково перевіряє прапорець is_admin у сесії.
- Валідація вхідних даних: На серверній стороні здійснюється базова валідація вхідних даних для всіх POST та PUT запитів, що допомагає запобігти збереженню неповних або некоректних даних та захищає від деяких видів атак.

Ці заходи безпеки забезпечують надійний захист даних користувачів та функціоналу веб-магазину від поширених загроз.

3.3 Тестування та налагодження

Етап тестування та налагодження є невід'ємною частиною процесу розробки програмного забезпечення, що дозволяє виявити та усунути помилки, забезпечити відповідність системи заданим вимогам та гарантувати її стабільну

роботу. Для веб-магазину спортивних добавок було проведено комплексне тестування, яке включало як ручне тестування функціоналу, так і налагодження виявлених проблем.

Тестування функціоналу здійснювалося шляхом імітації дій кінцевих користувачів та адміністраторів. Перевірялася коректність роботи всіх ключових функцій: реєстрації та авторизації (включаючи перевірку ролей), перегляду каталогу товарів з використанням пошуку, фільтрів та сортування, додавання товарів до кошика, зміни кількості та видалення, а також повного циклу оформлення замовлення з різними варіантами доставки та оплати. Особлива увага приділялася перевірці коректності відображення історії замовлень для користувачів та функціоналу адміністративної панелі (додавання, редагування, видалення продуктів; зміна статусів замовлень та повідомлень зворотного зв'язку).

Під час розробки та тестування було виявлено та успішно усунуто кілька критичних помилок. Однією з перших була проблема з підключенням до бази даних MySQL, яка проявлялася як "Host 'host.docker.internal' is not allowed to connect to this MySQL server". Ця помилка була вирішена шляхом надання відповідних привілеїв користувачеві MySQL на сервері бази даних, що дозволило Node.js додатку встановлювати з'єднання. Налагодження включало перевірку конфігурації користувачів MySQL та оновлення прав доступу.

Наступною значною проблемою стала TypeError: db.getConnection is not a function. Ця помилка виникла через некоректне використання об'єкта підключення до бази даних. Спочатку, файл db.js використовував mysql.createConnection(), який створює одне пряме підключення і не має методу getConnection(). Проблема була вирішена шляхом переходу на використання mysql.createPool() у db.js, що дозволило створити пул підключень, а також адаптації всіх викликів до бази даних у server.js для використання db.getConnection() та connection.release(). Це забезпечило правильну роботу з пулом підключень та підвищило ефективність управління ресурсами бази даних.

Крім того, було виявлено, що інтерактивні елементи в адміністративній панелі, такі як кнопки "редагувати", "видалити" та "додати новий продукт", не

реагували на кліки. Причиною виявилося те, що обробники подій прив'язувалися до елементів DOM до того, як ці елементи були динамічно згенеровані та вставлені в документ за допомогою JavaScript. Ця проблема була вирішена шляхом перенесення логіки прив'язки обробників подій таким чином, щоб вони виконувалися лише після повного рендерингу відповідних елементів інтерфейсу, зокрема після заповнення таблиць продуктів та замовлень. Також було перевірено коректність передачі даних між клієнтською та серверною частинами, забезпечуючи правильну валідацію та обробку вхідних даних на бекенді.

Процес налагодження включав використання інструментів розробника браузера для інспекції елементів та консолі JavaScript, а також логів сервера Node.js для відстеження помилок на бекенді. Послідовне виявлення та усунення цих помилок дозволило забезпечити стабільну та коректну роботу всіх компонентів веб-магазину.

У контексті створення веб-магазину, принципи моделювання та аналізу стохастичних шумових сигналів та інтелектуальної ідентифікації нелінійних систем можуть бути метафорично застосовані для розуміння та оптимізації роботи системи. Незважаючи на їхню пряму спрямованість на електроенергетику чи вимірювання, ці підходи дозволяють розробникам аналізувати та прогнозувати динамічні та непередбачувані "зовнішні впливи" на веб-додаток. Це включає коливання користувацького попиту, аномалії у поведінці системи, а також виявлення кіберзагроз, що дозволяє створювати більш адаптивні, стійкі та безпечні рішення. Таким чином, розуміння таких складних процесів дає інструменти для підвищення надійності та ефективності веб-магазину.

3.4 Висновок до третього розділу

У третьому розділі кваліфікаційної роботи було детально описано практичну реалізацію веб-магазину спортивних добавок, що охоплює як клієнтську, так і серверну частини, а також процес тестування та налагодження.

Було продемонстровано, як HTML використовувався для створення семантичної та логічної структури всіх сторінок, забезпечуючи основу для контенту. Створення CSS-стилів дозволило реалізувати сучасний та адаптивний дизайн, що забезпечує коректне відображення та зручність використання на різних пристроях.

Ключовим аспектом клієнтської частини стала розробка JavaScript-логіки, яка забезпечує динамічне завантаження контенту, інтерактивність користувацького інтерфейсу, управління станом кошика за допомогою Cookie, а також обробку всіх форм (реєстрація, вхід, оформлення замовлення, зворотний зв'язок). Особлива увага була приділена реалізації функціоналу адміністративної панелі, що дозволяє динамічно керувати продуктами, замовленнями та повідомленнями.

Реалізація серверної частини на базі Node.js та Express.js забезпечила надійний бекенд для обробки бізнес-логіки. Було налаштовано взаємодію з базою даних MySQL через пул підключень, що оптимізує продуктивність. Детально описано реалізацію всіх API-маршрутів, включаючи складні операції, такі як оформлення замовлення в рамках транзакції. Також були впроваджені важливі механізми безпеки, включаючи хешування паролів, управління сесіями та захист від SQL-ін'єкцій.

Процес тестування та налагодження відіграв вирішальну роль у забезпеченні стабільності системи. Були успішно вирішені проблеми з підключенням до бази даних, коректністю використання пулу підключень та функціональністю інтерактивних елементів в адмін-панелі. Всі виявлені помилки були усунені, що підтвердило готовність розробленої системи до функціонування. Таким чином, практична реалізація веб-магазину повністю відповідає поставленим вимогам та демонструє ефективне використання обраних технологій.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Долікарська допомога при кровотечах

Кровотеча є одним із найбільш небезпечних станів, що можуть виникнути внаслідок травм у побуті, на виробництві чи в інших умовах, і становить пряму загрозу життю людини. Вчасне та правильне надання долікарської допомоги має критичне значення, оскільки дозволяє зменшити крововтрату, запобігти розвитку шокового стану та врятувати життя постраждалого до прибуття кваліфікованої медичної допомоги [31]. Невміння або неправильні дії під час надання допомоги можуть значно погіршити стан людини, тому кожен громадянин повинен володіти базовими навичками зупинки кровотечі. Комплексний підхід до першої допомоги включає не лише безпосередню зупинку кровотечі, а й оцінку загального стану постраждалого, забезпечення його безпеки та виклик професійної медичної допомоги [33].

Залежно від типу пошкодженої судини, кровотечі поділяються на капілярні, венозні та артеріальні, кожна з яких вимагає специфічного підходу [32], [34].

Капілярна кровотеча. Виникає при пошкодженні найдрібніших кровоносних судин (капілярів). Кров виділяється повільно, невеликими краплями, по всій поверхні рани, нагадуючи росу. Така кровотеча рідко буває значною і, як правило, при відсутності проблем зі згортанням крові, зупиняється самостійно. Для її зупинки достатньо обробити рану антисептичним засобом (наприклад, перекисом водню або розчином антисептика) та накласти чисту, злегка тиснучу стерильну пов'язку [34]. Цей вид кровотечі є найменш загрозливим, проте важливо запобігти інфікуванню рани.

Венозна кровотеча. Характеризується безперервним, рівномірним витіканням крові темнішого, вишневого кольору, яка, на відміну від артеріальної, не пульсує [34]. При такій кровотечі існує ризик значної крововтрати, особливо при пошкодженні великих вен. Основним методом зупинки є накладання тиснучої пов'язки безпосередньо на рану. Для цього на рану кладуть стерильну серветку, а зверху — щільний валик з бинта чи тканини, який туго

прибинтовують. Важливо забезпечити достатній тиск, щоб зупинити потік крові, але не перетиснути кінцівку повністю. Якщо пошкоджено вену на кінцівці, їй необхідно надати підвищеного положення (вище рівня серця), що сприятиме зменшенню кровотечі завдяки гравітації [34]. У разі неефективності тиснучої пов'язки, можна застосувати прямий тиск на рану протягом 10-15 хвилин.

Артеріальна кровотеча. Є найбільш небезпечним видом, оскільки кров яскраво-червоного кольору витікає швидко, сильним пульсуючим струменем, синхронно з ударами серця [34]. Втрата значного об'єму крові відбувається за лічені хвилини, що створює безпосередню загрозу для життя постраждалого. Головним методом тимчасової зупинки масивної артеріальної кровотечі з кінцівок є накладання кровоспинного джгута (турнікета) [34], [35]. Джгут накладається на 5-7 см вище місця поранення, безпосередньо на одяг або на підкладену тканину (щоб не пошкодити шкіру та уникнути утиску нервів). Після накладання необхідно зафіксувати точний час (година та хвилини) у записці, яку слід закріпити на видному місці, оскільки час накладання джгута є критично важливим для запобігання некрозу тканин. Джгут можна безпечно тримати на кінцівці не більше 2 годин у теплу пору року та не більше 1 години – у холодну. Правильне застосування джгута рятує життя, але вимагає точного дотримання інструкцій [35].

У всіх випадках після надання першої допомоги необхідно негайно викликати бригаду екстреної (швидкої) медичної допомоги за номером 103, повідомивши диспетчеру про характер травми та стан постраждалого [31], [33]. До прибуття медиків необхідно контролювати стан постраждалого, забезпечити йому спокій та тепло, щоб запобігти переохолодженню та розвитку шоку. Важливо постійно спостерігати за диханням, свідомістю та кольором шкіри постраждалого.

4.2 Організація безпечних умов праці користувачів ПК

У сучасному цифровому світі комп'ютерна техніка є невід'ємною частиною робочого процесу в більшості професійних сфер. Тривала щоденна робота за

комп'ютером створює значне статичне та динамічне навантаження на організм людини. Це негативно впливає на органи зору, опорно-руховий апарат, серцево-судинну та нервову системи, що з часом може призводити до розвитку професійних захворювань, таких як синдром комп'ютерного зору, тунельний синдром зап'ястя, остеохондроз, головні болі та хронічна втома [36], [37]. Тому організація безпечних, здорових та комфортних умов праці для користувачів персональних комп'ютерів (ПК) є ключовим завданням охорони праці, що регулюється на законодавчому рівні.

Вимоги до організації роботи з екранними пристроями в Україні встановлюються «Вимогами щодо організації роботи з екранними пристроями», які затверджені Наказом Міністерства соціальної політики № 422 від 14.07.2021 [38]. Основні вимоги охоплюють декілька ключових аспектів, спрямованих на мінімізацію ризиків для здоров'я:

Робоче місце має бути спроєктоване так, щоб мінімізувати фізичне навантаження та забезпечити комфортну робочу позу. Правильна ергономіка передбачає, що робочий стіл повинен мати достатню площу для зручного розміщення обладнання (монітор, клавіатура, миша) та документів, дозволяючи вільно переміщувати руки та лікті. Робоче крісло має бути стійким, з регульованою висотою сидіння та спинки, а також з підлокітниками, щоб забезпечити підтримку поперекового відділу хребта та зменшити навантаження на руки [36], [37]. Монітор комп'ютера слід розташовувати на відстані 60-70 см від очей (довжина витягнутої руки), а його верхній край – на рівні або трохи нижче рівня очей. Це дозволяє уникнути напруження м'язів шиї та спини. Клавіатура та миша мають бути розташовані так, щоб руки знаходились у розслабленому положенні, а зап'ястя були прямими.

Недостатнє або неправильно організоване освітлення може спричиняти зорову втому, головний біль та зниження працездатності [36]. На робочому місці має бути забезпечене як природне, так і штучне освітлення, яке є рівномірним та не створює відблисків на екрані монітора та інших поверхнях. Слід уникати прямого потрапляння сонячного світла на екран. Рекомендується використовувати жалюзі або штори для регулювання природного освітлення.

Оптимальними параметрами мікроклімату в приміщенні є температура повітря в межах 20-25°С та відносна вологість 40-60% [37]. Важливо також забезпечити регулярне провітрювання приміщення для підтримки належної якості повітря.

Для запобігання перевтоми, пов'язаної з виконанням робіт за комп'ютером, необхідно встановлювати регламентовані перерви протягом робочого дня. Рекомендується робити короткі перерви тривалістю 5-10 хвилин після кожної години роботи [37]. Під час цих перерв доцільно відійти від комп'ютера, змінити позу та виконати спеціальні вправи для очей (наприклад, переведення погляду з близьких на віддалені предмети, кругові рухи очима, інтенсивне моргання) [37], [39]. Ці вправи допомагають зняти зорову напругу та покращити кровообіг. Також необхідно виконувати легкі фізичні вправи для розвантаження м'язів спини, шиї та плечового пояса (наприклад, нахили голови, оберти плечима, потягування). Регулярні фізичні паузи значно знижують ризик розвитку хронічних захворювань опорно-рухового апарату.

Дотримання цих вимог дозволяє не тільки запобігти розвитку професійних захворювань серед працівників, а й суттєво підвищити їхню продуктивність та створити сприятливий психологічний клімат у колективі. Свідоме ставлення до свого робочого місця є запорукою довготривалої працездатності та збереження здоров'я.

4.3 Висновок до четвертого розділу

У межах даного розділу було проаналізовано два ключові аспекти безпеки життєдіяльності та охорони праці, які мають фундаментальне значення в повсякденному житті та професійній діяльності. Розглянуто як заходи оперативного реагування на гострі небезпечні стани, так і превентивні заходи, спрямовані на збереження здоров'я в умовах сучасної праці.

Запропоновано дії та порядок надання долікарської допомоги при кровотечах, ці знання є обов’язковими у воєнний час. Володіння навичками, зокрема правилами накладання тиснучої пов'язки та джгута, є критично важливим для збереження життя постраждалого, адже своєчасні та правильні дії

дозволяють мінімізувати крововтрату до прибуття медичних працівників. Важливість швидкого реагування та чіткого виконання інструкцій у таких ситуаціях не можна переоцінити.

Розглянуто організацію безпечних умов праці для користувачів ПК, що є надзвичайно актуальним у зв'язку з тотальною цифровізацією більшості професій. Дотримання ергономічних вимог до робочого місця, норм освітлення, мікроклімату та режиму праці й відпочинку є обов’язковою умовою для профілактики професійних захворювань та збереження здоров'я працівників. Впровадження регулярних перерв та виконання спеціальних вправ для очей та тіла допомагає зберегти здоров’я та високу працездатність. Систематичний підхід до організації робочого простору має ключове значення для довгострокового добробуту.

Забезпечення безпеки життєдіяльності є комплексним завданням, що вимагає від людини як готовності до рішучих та грамотних дій в екстрених ситуаціях, так і свідомого, відповідального ставлення до організації свого робочого середовища відповідно до встановлених законодавчих норм і правил.

ВИСНОВКИ

У даній кваліфікаційній роботі було успішно розроблено та реалізовано повноцінний веб-магазин спортивних добавок із використанням сучасних веб-технологій. Проведена робота охопила всі ключові етапи життєвого циклу розробки програмного забезпечення, починаючи від глибокого аналізу предметної області та формування вимог, через детальне проектування архітектури та бази даних, і закінчуючи практичною реалізацією та тестуванням системи.

На етапі аналізу предметної області було вивчено сучасний ринок онлайн-продажів спортивного харчування, проаналізовано функціонал існуючих конкурентних рішень та визначено основні функціональні та нефункціональні вимоги до майбутнього веб-магазину. Це дозволило чітко сформулювати завдання проекту та обрати оптимальний технологічний стек, що включає HTML, CSS, JavaScript для клієнтської частини, Node.js з Express.js для серверної частини та MySQL як систему управління базами даних.

У проектній частині було розроблено архітектуру веб-додатку на основі клієнт-серверної моделі, детально описано схему взаємодії компонентів. Особливу увагу було приділено проектуванню реляційної бази даних, де були визначені сутності (users, products, orders, order_items, contacts), їхні зв'язки та обґрунтовано структуру кожної таблиці. Було також спроектовано RESTful API, що надає чітко визначені ендпоінти для всіх операцій, забезпечуючи ефективну комунікацію між фронтендом та бекендом. Проектування користувацького інтерфейсу та досвіду зосередилося на створенні інтуїтивно зрозумілого та адаптивного дизайну, що забезпечує зручність використання на різних пристроях.

Практична реалізація підтвердила життєздатність обраних технологій та архітектурних рішень. Клієнтська частина була розроблена з акцентом на динамічне відображення контенту, інтерактивність та зручність користувача, включаючи функціонал кошика на основі cookie та гнучкі форми замовлення. Серверна частина успішно реалізувала бізнес-логіку, взаємодію з базою даних

через пул підключень та обробку API-запитів, включаючи складні транзакційні операції для оформлення замовлень. Було впроваджено важливі механізми безпеки, такі як хешування паролів та управління сесіями, що забезпечує захист даних. Процес тестування та налагодження дозволив виявити та усунути низку критичних помилок, що підтвердило стабільність та коректність роботи системи.

Таким чином, всі поставлені мета та завдання кваліфікаційної роботи були успішно досягнуті. Створений веб-магазин є функціональним, безпечним та адаптивним рішенням, що відповідає сучасним вимогам електронної комерції. Отримані результати демонструють практичні навички у розробці повноцінних веб-додатків. Перспективи подальшого розвитку проекту включають розширення функціоналу адміністративної панелі, інтеграцію платіжних систем, додавання системи відгуків та рейтингів для продуктів, а також впровадження персоналізованих рекомендацій для користувачів.

ПЕРЕЛІК ДЖЕРЕЛ

1. Mozilla Developer Network (MDN) Web Docs. HTML (HyperText Markup Language) [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://developer.mozilla.org/uk/docs/Web/HTML (дата звернення: 16.02.2025).

2. Mozilla Developer Network (MDN) Web Docs. CSS (Cascading Style Sheets) [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://developer.mozilla.org/uk/docs/Web/CSS (дата звернення: 16.02.2025).

3. Mozilla Developer Network (MDN) Web Docs. JavaScript [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://developer.mozilla.org/uk/docs/Web/JavaScript (дата звернення: 17.02.2025).

4. Node.js. Official Documentation [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://nodejs.org/en/docs/ (дата звернення: 20.02.2025).

5. Express.js. Official Documentation [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://expressjs.com/ (дата звернення: 24.02.2025).

6. MySQL. Official Documentation [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://dev.mysql.com/doc/ (дата звернення: 21.02.2025).

7. W3C. HTML Living Standard [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://html.spec.whatwg.org/multipage/ (дата звернення: 15.02.2025).

8. W3C. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://www.w3.org/TR/CSS21/ (дата звернення: 16.02.2025).

9. FoxmindEd. UX дослідження: Що це і чому вони важливі? [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://foxminded.ua/ux-doslidzhennia-shcho-tse/ (дата звернення: 28.02.2025).

10. Stack Overflow. Questions & Answers for Developers [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://stackoverflow.com/ (дата звернення: 10.03.2025). (дата звернення: 10.03.2025).

11. GeeksforGeeks. A computer science portal for geeks [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/ (дата звернення: 11.03.2025).

12. FreeCodeCamp. Learn to code — for free [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://www.freecodecamp.org/ (дата звернення: 12.03.2025).

13. Css.in.ua. Український веб-довідник [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://css.in.ua/ (дата звернення: 12.03.2025).

14. Html-css.co.ua. HTML і CSS довідник українською [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://html-css.co.ua/ (дата звернення: 05.04.2025).

15. IT Школа Hillel. Архітектура веб-додатків - як вибрати? [Електронний ресурс] – 2022. – Режим доступу до ресурсу: https://blog.ithillel.ua/articles/web-application-architecture (дата звернення: 23.02.2025).

16. ITGid.info. Курси Олександра Лущенка (включно з Node.js та MySQL) [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://itgid.info/ua/ (дата звернення: 22.02.2025).

17. W3Schools. Web Development Resources [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://www.w3schools.com/ (дата звернення: 10.03.2025).

18. DigitalOcean. Tutorials for Developers [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://www.digitalocean.com/community/tutorials (дата звернення: 15.03.2025).

19. Web.dev. Learn the latest web development best practices [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://web.dev/ (дата звернення: 29.02.2025).

20. SQL.org. The SQL Standard [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://sql.org/ (дата звернення: 21.02.2025).

21. GoIT. REST API: що це, як працює, переваги і приклади [Електронний ресурс] – 2024. – Режим доступу до ресурсу:

https://goit.global/ua/articles/rest-api-shcho-tse-iak-pratsiuie-perevahy-i-pryklady/ (дата звернення: 18.03.2025).

22. Robot_dreams. Вступ до REST API — RESTful вебсервіси [Електронний ресурс] – 2024. – Режим доступу до ресурсу: https://robotdreams.cc/uk/blog/466-vstup-do-rest-api-restful-vebservisi (дата звернення: 18.03.2025).

23. Microsoft Support. Посібник зі зв’язків між таблицями [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://support.microsoft.com/uk-ua/topic/посібник-зі-зв-язків-між-таблицями-30446197-4fbe-457b-b992-2f6fb812b58f (дата звернення: 24.02.2025).

24. Dijix. Зв’язки між таблицями в Sql [Електронний ресурс] – 2023. – Режим доступу до ресурсу: https://dijix.com.ua/blog/uk/svyazi-mezhdu-tabliczami-v-sql/ (дата звернення: 25.02.2025).

25. Blog.postman. Що таке HTTP-методи? [Електронний ресурс] – 2023. – Режим доступу до ресурсу: https://blog.postman.com/what-are-http-methods/ (дата звернення: 10.03.2025).

26. Drukarnia. Хешування паролів [Електронний ресурс] – 2025. – Режим доступу до ресурсу: https://drukarnia.com.ua/articles/kheshuvannya-paroliv-vikoristannya-soli-ta-bcrypt-fsme- (дата звернення: 08.04.2025).

27. Coi. Тренди екомерс [Електронний ресурс] – 2024. – Режим доступу до ресурсу: https://coi.ua/blog/AdMarketing/e-commerce-trends-in-2024/ (дата звернення: 25.02.2025).

28. Pro consulting. Аналіз ринку спортивного харчування в Україні [Електронний ресурс] – 2024. – Режим доступу до ресурсу: https://pro-consulting.ua/ua/issledovanie-rynka/analiz-rynka-sportivnogo-pitaniya-v-ukraine-2024-god (дата звернення: 24.02.2025).

29. DSN Group. Дослідження ринку добавок і вітамінів в Україні [Електронний ресурс] – 2024. – Режим доступу до ресурсу: https://dsn.com.ua/doslidzhennia-rynku-dobavok-i-vitaminiv-v-ukraini/ (дата звернення: 26.02.2025).

30. Iampm. Технології для розробки сайту: що це, які найпопулярніші та як вибрати [Електронний ресурс] – 2024. – Режим доступу до ресурсу: https://iampm.club/ua/blog/tehnologiyi-dlya-rozrobki-sajtu-shho-cze-yaki-najpopulyarnishi-ta-yak-vibrati/ (дата звернення: 23.02.2025).

31. Порядок надання домедичної допомоги постраждалим при кровотечах та шоці. Наказ МОЗ України від 16.06.2014 № 398 [Електронний ресурс] – 2024. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0750-14 (дата звернення: 15.04.2025).

32. Види кровотеч, методи зупинки кровотечі. Alpha Cell Clinic. [Електронний ресурс] – 2022. – Режим доступу до ресурсу: https://redcross.org.ua/fat/guide-firstaid/ (дата звернення: 16.04.2025).

33. Товариство Червоного Хреста України. Перша допомога при кровотечах [Електронний ресурс] – 2024. – Режим доступу до ресурсу: https://redcross.org.ua/fahova-diyalnist/persha-dopomoga/a-krovotechi/ (дата звернення: 16.04.2025).

34. Крилюк В. О., та ін. Домедична допомога на місці події: практичний посібник. – Київ: Медицина, 2020. – 88 с. [Електронний ресурс] – 2020. – Режим доступу до ресурсу: https://dsns.gov.ua/upload/9/5/7/9/2020-5-19-112-posibnik.pdf (дата звернення: 17.04.2025).

35. FAST. Зупинка кровотеч. Накладання турнікету [Електронний ресурс] – 2023. – Режим доступу до ресурсу: https://ingeniusua.org/articles/tymchasova-zupynka-krovotechi-pravyla-nakladannya-dzhhuta (дата звернення: 18.04.2025).

36. Основи охорони праці користувачів персональних комп'ютерів : навчальний посібник для студентів ВНЗ та інженерів-практиків. Видавництво «Богдан». [Електронний ресурс] – 2024. – Режим доступу до ресурсу: https://bohdan-books.com/catalog/book/105263/ (дата звернення: 20.04.2025).

37. Гігієна праці в деталях: вимоги при роботі з екранними пристроями. Держпраці. [Електронний ресурс] – 2024. – Режим доступу до ресурсу: https://pd.dsp.gov.ua/news/hihiiena-pratsi-v-detaliakh-vymohy-pry-roboti-z-ekrannymy-prystroiamy/ (дата звернення: 21.04.2025).

38. Вимоги щодо організації роботи з екранними пристроями. Затверджено Наказом Міністерства соціальної політики України від 14.07.2021 № 422 [Електронний ресурс] – 2021. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z1029-21 (дата звернення: 22.04.2025).

39. Вправи для очей: збереження здорових очей при роботі на комп'ютері. Бажаємо здоров'я. [Електронний ресурс] – 2023. – Режим доступу до ресурсу: https://apteka.net.ua/articles/vpravy-dlya-ochey-zberezhennya-zdorovykh-ochey-pry-roboti-na-kompyuteri (дата звернення: 23.04.2025).

40. Stochastic Noise, Signal Identification and Dynamic Systems: In honor of Professor Oleg Vasylenko’s 60th Anniversary [Електронний ресурс] – 2024. – Режим доступу до ресурсу: https://doi.org/10.1007/978-3-031-71093-3 (дата звернення: 05.06.2025).

41. Intellectual Identification of Nonlinear Systems with Stochastic Noise [Електронний ресурс] – 2024. – Режим доступу до ресурсу: https://doi.org/10.1007/978-3-031-82035-9_7 (дата звернення: 09.06.2025).

ДОДАТКИ

Додаток А

Код розробленого інтернет магазину

script.js

```
document.addEventListener('DOMContentLoaded', () => {
    // Елементи DOM, які можуть бути на різних сторінках
    const productListContainer = document.getElementById('product-
list-container');
    const searchInput = document.getElementById('search-input');
    const searchButton = document.getElementById('search-button');

    const loginForm = document.getElementById('login-form');
    const authLink = document.getElementById('auth-link');
    const adminLink = document.getElementById('admin-link');
    const userOrdersLink = document.getElementById('my-orders-nav-
link');

    const registerForm = document.getElementById('register-form');
    const contactForm = document.getElementById('contact-form');

    // Елементи адмін-панелі
    const productsListDiv = document.getElementById('products-
list');
    const ordersListContainer = document.getElementById('orders-
list');
    const contactsListContainer =
document.getElementById('contacts-list');
    const userOrdersContainer = document.getElementById('user-
orders-list');

    // Елементи кошика
    const cartItemsContainer = document.getElementById('cart-
items');
    const cartTotalSpan = document.getElementById('cart-total');
    const checkoutForm = document.getElementById('checkout-form');

    // Елементи доставки
    const deliveryTypeSelect = document.getElementById('delivery-
type');
    const novaPoshtaDetailsDiv = document.getElementById('nova-
poshta-details');
    const homeAddressDetailsDiv = document.getElementById('home-
address-details');
    const novaPoshtaWarehouseInput = document.getElementById('nova-
poshta-warehouse');
    const fullAddressInput = document.getElementById('full-
address');
    const customerPhoneInput = document.getElementById('customer-
phone');

    // Елементи фільтрів
```

```
    const toggleSidebarBtn = document.getElementById('toggle-
sidebar-btn');
    const filtersSidebar = document.getElementById('filters-
sidebar');
    const mainContentWithSidebar =
document.querySelector('main.has-sidebar');

    // Функції для роботи з кошиком (через cookie)
    function getCartFromCookie() {
        const cookies = document.cookie.split(';');
        for (let i = 0; i < cookies.length; i++) {
            let cookie = cookies[i].trim();
            if (cookie.startsWith('cart=')) {
                try {
                    return JSON.parse(cookie.substring(5));
                } catch (e) {
                    console.error('Помилка парсингу куки кошика:',
e);
                    return {};
                }
            }
        }
        return {};
    }

    function saveCartToCookie(cart) {
        document.cookie = `cart=${JSON.stringify(cart)};path=/;max-
age=${60 * 60 * 24 * 30}`;
    }

    function addProductToCart(productId) {
        const cart = getCartFromCookie();
        cart[productId] = (cart[productId] || 0) + 1;
        saveCartToCookie(cart);
        alert('Товар додано!'); // Скорочено
        updateCartCount();
    }

    function removeProductFromCart(productId) {
        const cart = getCartFromCookie();
        delete cart[productId];
        saveCartToCookie(cart);
        renderCartItems();
        updateCartCount();
    }

    function updateCartItemQuantity(productId, newQuantity) {
        const cart = getCartFromCookie();
        if (newQuantity <= 0) {
            removeProductFromCart(productId);
            return;
        }
        cart[productId] = newQuantity;
        saveCartToCookie(cart);
        renderCartItems();
```

```
        updateCartCount();
    }
    function clearCart() {
        saveCartToCookie({});
        renderCartItems();
        updateCartCount();
    }
    function updateCartCount() {
        const cart = getCartFromCookie();
        const totalItems = Object.values(cart).reduce((sum,
quantity) => sum + quantity, 0);
        const cartCountElement = document.getElementById('cart-
count');
        if (cartCountElement) cartCountElement.textContent =
totalItems;
        if (cartTotalSpan) renderCartItems();
    }
    async function fetchCartItemsData() {
        const cart = getCartFromCookie();
        const productIds = Object.keys(cart).map(Number);
        if (productIds.length === 0) return [];
        try {
            const res = await fetch('/api/cart/items', { method:
'POST', headers: { 'Content-Type': 'application/json' }, body:
JSON.stringify({ productIds }) });
            if (!res.ok) {
                const errData = await res.json();
                throw new Error(errData.error || 'Помилка кошика.');
// Скорочено
            }
            const productsData = await res.json();
            return productsData.map(p => ({ ...p, quantity:
cart[p.id] }));
        } catch (error) {
            console.error('Помилка завантаження кошика:', error);
            alert('Помилка кошика. Спробуйте пізніше.'); //
Скорочено
            return [];
        }
    }
    async function renderCartItems() {
        if (!cartItemsContainer || !cartTotalSpan) return;
        const items = await fetchCartItemsData();
        cartItemsContainer.innerHTML = '';
        let total = 0;
        if (items.length === 0) {
            cartItemsContainer.innerHTML = '<p>Кошик
порожній.</p>'; // Скорочено
            cartTotalSpan.textContent = '0.00 грн';
            return;
        }
        items.forEach(item => {
            const itemTotal = item.price * item.quantity;
            total += itemTotal;
            const cartItemDiv = document.createElement('div');
```

```
            cartItemDiv.classList.add('cart-item');
            cartItemDiv.innerHTML = `
                <img src="${item.image_url}" alt="${item.name}">
                <div class="item-details">
                    <h4>${item.name}</h4>
                    <p>Ціна: ${item.price.toFixed(2)} грн</p>
                    <div class="quantity-controls">
                        <button   class="decrease-quantity"   data-
product-id="${item.id}">-</button>
                        <span>${item.quantity}</span>
                        <button   class="increase-quantity"   data-
product-id="${item.id}">+</button>
                    </div>
                    <p>Загалом: ${(itemTotal).toFixed(2)} грн</p>
                    <button   class="remove-from-cart-btn"   data-
product-id="${item.id}">Видалити</button>
                </div>
            `;
            cartItemsContainer.appendChild(cartItemDiv);
        });

        cartTotalSpan.textContent = `${total.toFixed(2)} грн`;

        // Додавання обробників подій для кнопок кошика
        cartItemsContainer.querySelectorAll('.increase-
quantity').forEach(btn => btn.addEventListener('click', (e) => {
            const pId = parseInt(e.target.dataset.productId);
            updateCartItemQuantity(pId, getCartFromCookie()[pId] +
1);
        }));
        cartItemsContainer.querySelectorAll('.decrease-
quantity').forEach(btn => btn.addEventListener('click', (e) => {
            const pId = parseInt(e.target.dataset.productId);
            updateCartItemQuantity(pId, getCartFromCookie()[pId] -
1);
        }));
        cartItemsContainer.querySelectorAll('.remove-from-cart-
btn').forEach(btn => btn.addEventListener('click', (e) => {

removeProductFromCart(parseInt(e.target.dataset.productId));
        }));
    }
    // Завантаження та відображення продуктів
    async function fetchProducts(filters = {}, isHomePage = false)
{
        if (!productListContainer) return;
        let apiUrl = '/api/products';
        if (isHomePage) apiUrl = '/api/popular-products';
        else       {       const       queryString       =       new
URLSearchParams(filters).toString(); apiUrl += `?${queryString}`; }
        try {
            const res = await fetch(apiUrl);
            if (!res.ok) {
                const errData = await res.json();
```

```
                throw new Error(`Помилка: ${res.status} -
${errData.error || 'Невідомо'}`); // Скорочено
            }
            const products = await res.json();
            productListContainer.innerHTML = '';
            if (products.length === 0) {
                productListContainer.innerHTML = '<p>Продукти не
знайдено.</p>'; // Скорочено
                return;
            }
            products.forEach(product => {
                const card = document.createElement('div');
                card.classList.add('product-card');
                card.innerHTML = `
                    <a href="/product.html?id=${product.id}"
class="product-card-link">
                        <img src="${product.image_url}"
alt="${product.name}">
                        <h3>${product.name}</h3>
                        <div class="product-details">
                            <ul>
                                <li>Тип: ${product.type ||
'N/A'}</li>
                                <li>Вага: ${product.weight_grams ?
product.weight_grams + ' г' : 'N/A'}</li>
                                <li>Виробник:
${product.manufacturer || 'N/A'}</li>
                            </ul>
                        </div>
                        <p
class="price">${product.price.toFixed(2)} грн</p>
                    </a>
                    <button class="add-to-cart-btn" data-product-
id="${product.id}">Додати до кошика</button>
                `;
                productListContainer.appendChild(card);
            });
            document.querySelectorAll('.add-to-cart-
btn').forEach(btn => btn.addEventListener('click', (e) =>
addProductToCart(parseInt(e.target.dataset.productId))));
        } catch (error) {
            console.error('Помилка завантаження продуктів:',
error);
            if (productListContainer)
productListContainer.innerHTML = `<p>Помилка завантаження:
${error.message}</p>`; // Скорочено
        }
    }

    // Завантаження деталей продукту
    async function fetchProductDetail() {
        const productDetailContainer =
document.getElementById('product-detail');
        const pId = new
URLSearchParams(window.location.search).get('id');
```

```
        if (!productDetailContainer || !pId) return;
        try {
            const res = await fetch(`/api/products/${pId}`);
            const product = await res.json();
            if (!product) { productDetailContainer.innerHTML =
'<p>Продукт не знайдено.</p>'; return; }
            document.title = `${product.name} - IronLab`;
            productDetailContainer.innerHTML = `
                <div class="product-detail-card">
                    <img src="${product.image_url}"
alt="${product.name}" class="product-detail-image">
                    <div class="product-info">
                        <h2>${product.name}</h2>
                        <p class="product-
description">${product.description}</p>
                        <p class="product-price">Ціна:
<span>${product.price.toFixed(2)} грн</span></p>
                        <p>Тип: ${product.type}</p>
                        <p>Виробник: ${product.manufacturer}</p>
                        <p>Вага: ${product.weight_grams ?
product.weight_grams + ' г' : 'N/A'}</p>
                        <button class="add-to-cart-btn" data-
product-id="${product.id}">Додати до кошика</button>
                    </div>
                </div>
            `;
            productDetailContainer.querySelector('.add-to-cart-
btn').addEventListener('click', (e) =>
addProductToCart(parseInt(e.target.dataset.productId)));
        } catch (error) {
            console.error('Помилка завантаження продукту:', error);
            productDetailContainer.innerHTML = '<p>Помилка
завантаження продукту.</p>'; // Скорочено
        }
    }

    // Перевірка аутентифікації та відображення посилань
    async function checkAuth() {
        try {
            const res = await fetch('/api/check-auth');
            const data = await res.json();
            if (data.isAuthenticated) {
                if (authLink) { authLink.innerHTML = `<a
href="#">Вийти (${data.username})</a>`;
authLink.querySelector('a').addEventListener('click',
handleLogout); }
                if (userOrdersLink) userOrdersLink.style.display =
'list-item';
                if (data.isAdmin && adminLink)
adminLink.style.display = 'list-item';
            } else {
                if (authLink) authLink.innerHTML = '<a
href="/login.html">Увійти</a>';
                if (userOrdersLink) userOrdersLink.style.display =
'none';
```

```
                if (adminLink) adminLink.style.display = 'none';
            }
        } catch (error) {
            console.error('Помилка автентифікації:', error);
            // Залишаємо посилання як для неавторизованого
            if (authLink) authLink.innerHTML = '<a
href="/login.html">Увійти</a>';
            if (userOrdersLink) userOrdersLink.style.display =
'none';
            if (adminLink) adminLink.style.display = 'none';
        }
    }

    // Обробка входу
    async function handleLogin(e) {
        e.preventDefault();
        const email = document.getElementById('login-email').value;
        const password = document.getElementById('login-
password').value;
        try {
            const res = await fetch('/api/login', { method: 'POST',
headers: { 'Content-Type': 'application/json' }, body:
JSON.stringify({ email, password }) });
            const result = await res.json();
            if (res.ok) { alert(result.message);
window.location.href = '/'; }
            else alert(`Помилка входу: ${result.error}`);
        } catch (error) {
            console.error('Помилка мережі при вході:', error);
            alert('Помилка входу. Спробуйте пізніше.'); // Скорочено
        }
    }

    // Обробка реєстрації
    async function handleRegister(e) {
        e.preventDefault();
        const username = document.getElementById('username').value;
        const email = document.getElementById('register-
email').value;
        const password = document.getElementById('register-
password').value;
        const confirmPassword = document.getElementById('confirm-
password').value;
        if (password !== confirmPassword) { alert('Паролі не
співпадають!'); return; }
        try {
            const res = await fetch('/api/register', { method:
'POST', headers: { 'Content-Type': 'application/json' }, body:
JSON.stringify({ username, email, password }) });
            const result = await res.json();
            if (res.ok) { alert(result.message);
window.location.href = '/login.html'; }
            else alert(`Помилка реєстрації: ${result.error}`);
        } catch (error) {
            console.error('Помилка мережі при реєстрації:', error);
```

```
            alert('Помилка реєстрації. Спробуйте пізніше.'); // Скорочено
        }
    }

    // Обробка виходу
    async function handleLogout(e) {
        e.preventDefault();
        try {
            const res = await fetch('/api/logout', { method: 'POST' });
            const result = await res.json();
            if (res.ok) { alert(result.message); window.location.href = '/login.html'; }
            else alert(`Помилка виходу: ${result.error}`);
        } catch (error) {
            console.error('Помилка мережі при виході:', error);
            alert('Помилка виходу. Спробуйте пізніше.'); // Скорочено
        }
    }

    // Функції адмін-панелі: Продукти
    async function fetchAdminProducts() {
        if (!productsListDiv) return;
        try {
            const res = await fetch('/api/products');
            const products = await res.json();
            productsListDiv.innerHTML = '';
            if (products.length === 0) { productsListDiv.innerHTML = '<p>Немає продуктів.</p>'; return; } // Скорочено

            const table = document.createElement('table');
            table.classList.add('products-admin-table');
            table.innerHTML = `
                <thead><tr><th>ID</th><th>Назва</th><th>Ціна</th><th>Тип</th><th>Виробник</th><th>Дії</th></tr></thead><tbody></tbody>`; // Спрощена таблиця
            const tbody = table.querySelector('tbody');
            products.forEach(p => {
                const row = tbody.insertRow();
                row.innerHTML = `
                    <td>${p.id}</td>
                    <td>${p.name}</td>
                    <td>${p.price.toFixed(2)}</td>
                    <td>${p.type}</td>
                    <td>${p.manufacturer}</td>
                    <td class="actions">
                        <button class="edit-product-btn" data-product-id="${p.id}">Редагувати</button>
                        <button class="delete-product-btn" data-product-id="${p.id}">Видалити</button>
                    </td>
                `;
```

```
            });
            productsListDiv.appendChild(table);
            document.querySelectorAll('.edit-product-
btn').forEach(btn    =>    btn.addEventListener('click',    (e)    =>
editProduct(e.target.dataset.productId)));
            document.querySelectorAll('.delete-product-
btn').forEach(btn => btn.addEventListener('click', deleteProduct));
        } catch (error) {
            console.error('Помилка завантаження адмін-продуктів:',
error);
            if    (productsListDiv)    productsListDiv.innerHTML    =
'<p>Помилка завантаження продуктів.</p>'; // Скорочено
        }
    }

    async function editProduct(productId) {
        const  pEditForm  =  document.getElementById('product-edit-
form');
        const  pFormTitle  =  document.getElementById('product-form-
title');
        const   saveProductBtn   =   document.getElementById('save-
product-btn');
        const pIdField = document.getElementById('product-id');
        const pForm = document.getElementById('product-form');

        if (pForm) pForm.reset();
        if (pIdField) pIdField.value = '';
        const   pWeightInput   =   document.getElementById('product-
weight-grams');
        if (pWeightInput) pWeightInput.value = '';

        if (productId) {
            if (pFormTitle) pFormTitle.textContent = 'Редагувати';
// Скорочено
            if    (saveProductBtn)    saveProductBtn.textContent    =
'Зберегти'; // Скорочено
            try {
                const            res            =            await
fetch(`/api/products/${productId}`);
                const product = await res.json();
                if (res.ok) {
                    if (pIdField) pIdField.value = product.id;
                    if    (document.getElementById('product-name'))
document.getElementById('product-name').value = product.name;
                    if           (document.getElementById('product-
description')) document.getElementById('product-description').value
= product.description;
                    if   (document.getElementById('product-price'))
document.getElementById('product-price').value = product.price;
                    if    (document.getElementById('product-type'))
document.getElementById('product-type').value = product.type;
                    if           (document.getElementById('product-
manufacturer'))                  document.getElementById('product-
manufacturer').value = product.manufacturer;
```

```
                    if     (document.getElementById('product-image-
url'))     document.getElementById('product-image-url').value     =
product.image_url;
                    if    (pWeightInput)    pWeightInput.value    =
product.weight_grams || '';
                    if    (pEditForm)    pEditForm.style.display    =
'block';
                } else alert(`Помилка: ${product.error}`);
            } catch (error) {
                console.error('Помилка  завантаження  для  ред.:',
error); // Скорочено
                alert('Помилка    завантаження    продукту.');    //
Скорочено
            }
        } else {
            if (pFormTitle) pFormTitle.textContent = 'Додати'; //
Скорочено
            if    (saveProductBtn)    saveProductBtn.textContent    =
'Додати'; // Скорочено
            if (pEditForm) pEditForm.style.display = 'block';
        }
    }

    async function saveProduct(e) {
        e.preventDefault();
        const   pId   =   document.getElementById('product-id')   ?
document.getElementById('product-id').value : null;
        const name = document.getElementById('product-name').value;
        const   description   =   document.getElementById('product-
description').value;
        const price = parseFloat(document.getElementById('product-
price').value);
        const type = document.getElementById('product-type').value;
        const   manufacturer   =   document.getElementById('product-
manufacturer').value;
        const  weight  =  document.getElementById('product-weight-
grams');
        const weight_grams = weight ? (parseFloat(weight.value) ||
null) : null;
        const  imageUrl  =  document.getElementById('product-image-
url').value;

        const   pData   =   {   name,   description,   price,   type,
manufacturer, image_url: imageUrl, weight_grams };

        try {
            let res;
            if (pId) res = await fetch(`/api/admin/products/${pId}`,
{ method: 'PUT', headers: { 'Content-Type': 'application/json' },
body: JSON.stringify(pData) });
            else res = await fetch('/api/admin/products', { method:
'POST',  headers:  {  'Content-Type':  'application/json'  },  body:
JSON.stringify(pData) });

            const result = await res.json();
```

```
            if     (res.ok)     {     alert(result.message);     if
(document.getElementById('product-edit-form'))
document.getElementById('product-edit-form').style.display        =
'none'; fetchAdminProducts(); fetchProducts(); }
            else alert(`Помилка: ${result.error || 'Невідомо'}`); //
Скорочено
        } catch (error) {
            console.error('Помилка збереження продукту:', error);
            alert('Помилка збереження продукту.'); // Скорочено
        }
    }

    async function deleteProduct(e) {
        const pId = e.target.dataset.productId;
        if  (!confirm(`Видалити  продукт  #${pId}?`))  return;  //
Скорочено

        try {
            const res = await fetch(`/api/admin/products/${pId}`, {
method: 'DELETE' });
            const result = await res.json();
            if       (res.ok)       {       alert(result.message);
fetchAdminProducts(); fetchProducts(); }
            else alert(`Помилка: ${result.error || 'Невідомо'}`); //
Скорочено
        } catch (error) {
            console.error('Помилка видалення продукту:', error);
            alert('Помилка видалення продукту.'); // Скорочено
        }
    }

    // Функції адмін-панелі: Замовлення
    async function fetchAdminOrders() {
        if (!ordersListContainer) return;
        try {
            const res = await fetch('/api/admin/orders');
            if   (!res.ok)   {   ordersListContainer.innerHTML   =
res.status   ===   403   ?   '<p>Немає   прав.</p>'   :   '<p>Помилка
замовлень.</p>'; return; } // Скорочено
            const orders = await res.json();
            ordersListContainer.innerHTML = '';
            if (orders.length === 0) { ordersListContainer.innerHTML
= '<p>Немає замовлень.</p>'; return; } // Скорочено

            const table = document.createElement('table');
            table.classList.add('orders-table');
            table.innerHTML = `

<thead><tr><th>ID</th><th>Клієнт</th><th>Сума</th><th>Статус</th><
th>Дата</th><th>Дії</th></tr></thead><tbody></tbody>`; // Спрощена
таблиця
            const tbody = table.querySelector('tbody');
            orders.forEach(order => {
                const row = tbody.insertRow();
                row.innerHTML = `
```

```
                    <td>${order.id}</td>
                    <td>${order.customer_name}</td>
                    <td>${order.total_amount.toFixed(2)} грн</td>
                    <td>
                        <select class="order-status-select" data-
order-id="${order.id}">
                            <option value="Pending" ${order.status
=== 'Pending' ? 'selected' : ''}>В обробці</option>
                            <option value="Completed"
${order.status === 'Completed' ? 'selected' : ''}>Завершено</option>
                            <option value="Cancelled"
${order.status === 'Cancelled' ? 'selected' : ''}>Скасовано</option>
                        </select>
                    </td>
                    <td>${new
Date(order.created_at).toLocaleString()}</td>
                    <td><button class="save-order-status-btn" data-
order-id="${order.id}">Зберегти</button></td>
                `;
            });
            ordersListContainer.appendChild(table);
            document.querySelectorAll('.save-order-status-
btn').forEach(btn => btn.addEventListener('click', async (e) => {
                const oId = e.target.dataset.orderId;
                const statusSelect =
document.querySelector(`.order-status-select[data-order-
id="${oId}"]`);
                await updateOrderStatus(oId, statusSelect.value);
            }));
        } catch (error) {
            console.error('Помилка завантаження адмін-замовлень:',
error);
            if (ordersListContainer) ordersListContainer.innerHTML
= '<p>Помилка завантаження замовлень.</p>'; // Скорочено
        }
    }

    async function updateOrderStatus(orderId, newStatus) {
        try {
            const res = await
fetch(`/api/admin/orders/${orderId}/status`, { method: 'PUT',
headers: { 'Content-Type': 'application/json' }, body:
JSON.stringify({ status: newStatus }) });
            const result = await res.json();
            if (res.ok) { alert(result.message);
fetchAdminOrders(); }
            else alert(`Помилка: ${result.error || 'Невідомо'}`); //
Скорочено
        } catch (error) {
            console.error('Помилка оновлення статусу:', error);
            alert('Помилка оновлення статусу.'); // Скорочено
        }
    }

    // Функції адмін-панелі: Контакти
```

```
    async function fetchAdminContacts() {
        if (!contactsListContainer) return;
        try {
            const res = await fetch('/api/admin/contacts');
            if (!res.ok) { contactsListContainer.innerHTML = res.status === 403 ? '<p>Немає прав.</p>' : '<p>Помилка повідомлень.</p>'; return; } // Скорочено
            const messages = await res.json();
            contactsListContainer.innerHTML = '';
            if (messages.length === 0) { contactsListContainer.innerHTML = '<p>Немає повідомлень.</p>'; return; } // Скорочено

            const table = document.createElement('table');
            table.classList.add('contacts-table');
            table.innerHTML = `
<thead><tr><th>ID</th><th>Ім'я</th><th>Email</th><th>Статус</th><th>Дії</th></tr></thead><tbody></tbody>`; // Спрощена таблиця
            const tbody = table.querySelector('tbody');
            messages.forEach(msg => {
                const row = tbody.insertRow();
                row.innerHTML = `
                    <td>${msg.id}</td>
                    <td>${msg.name}</td>
                    <td>${msg.email}</td>
                    <td>
                        <select class="contact-status-select" data-contact-id="${msg.id}">
                            <option value="New" ${msg.status === 'New' ? 'selected' : ''}>Нове</option>
                            <option value="Resolved" ${msg.status === 'Resolved' ? 'selected' : ''}>Вирішено</option>
                        </select>
                    </td>
                    <td><button class="save-contact-status-btn" data-contact-id="${msg.id}">Зберегти</button></td>
                `;
            });
            contactsListContainer.appendChild(table);
            document.querySelectorAll('.save-contact-status-btn').forEach(btn => btn.addEventListener('click', async (e) => {
                const cId = e.target.dataset.contactId;
                const statusSelect = document.querySelector(`.contact-status-select[data-contact-id="${cId}"]`);
                await updateContactStatus(cId, statusSelect.value);
            }));
        } catch (error) {
            console.error('Помилка завантаження адмін-повідомлень:', error);
            if (contactsListContainer) contactsListContainer.innerHTML = '<p>Помилка завантаження повідомлень.</p>'; // Скорочено
        }
```

```
    }

    async function updateContactStatus(contactId, newStatus) {
        try {
            const res = await fetch(`/api/admin/contacts/${contactId}/status`, { method: 'PUT', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ status: newStatus }) });
            const result = await res.json();
            if (res.ok) { alert(result.message); fetchAdminContacts(); }
            else alert(`Помилка: ${result.error || 'Невідомо'}`); // Скорочено
        } catch (error) {
            console.error('Помилка оновлення статусу повідомлення:', error);
            alert('Помилка оновлення статусу повідомлення.'); // Скорочено
        }
    }

    // Замовлення користувача
    async function fetchUserOrders() {
        if (!userOrdersContainer) return;
        try {
            const res = await fetch('/api/my-orders');
            if (!res.ok) { userOrdersContainer.innerHTML = res.status === 401 ? '<p><a href="/login.html">Увійдіть</a>, щоб переглянути.</p>' : '<p>Помилка замовлень.</p>'; return; } // Скорочено
            const orders = await res.json();
            userOrdersContainer.innerHTML = '';
            if (orders.length === 0) { userOrdersContainer.innerHTML = '<p>У вас ще немає замовлень.</p>'; return; } // Скорочено

            orders.forEach(order => {
                const card = document.createElement('div');
                card.classList.add('order-card');
                card.innerHTML = `
                    <h3>Замовлення #${order.id}</h3>
                    <p>Статус: <span class="order-status ${order.status.toLowerCase()}">${order.status}</span></p>
                    <p>Сума: ${order.total_amount.toFixed(2)} грн</p>
                    <p>Дата: ${new Date(order.created_at).toLocaleString()}</p>
                    <p>Адреса: ${order.shipping_address}</p>
                    <h4>Товари:</h4>
                    <ul>
                        ${order.items.map(item => `<li>${item.product_name} x ${item.quantity}</li>`).join('')}
                    </ul>
                `; // Спрощені деталі
                userOrdersContainer.appendChild(card);
            });
```

```
        } catch (error) {
            console.error('Помилка завантаження замовлень користувача:', error);
            if (userOrdersContainer) userOrdersContainer.innerHTML = '<p>Помилка завантаження замовлень.</p>'; // Скорочено
        }
    }

    // Ініціалізація та обробники подій
    checkAuth();
    updateCartCount();

    // Головна сторінка
    if (document.body.classList.contains('homepage') && productListContainer) fetchProducts({}, true);

    // Сторінка продуктів
    if (document.body.classList.contains('products-page') && productListContainer) {
        const getCurrFilters = () => {
            const search = searchInput ? searchInput.value : '';
            const types = Array.from(document.querySelectorAll('#type-filter-group input:checked')).map(cb => cb.value);
            const manufacturers = Array.from(document.querySelectorAll('#manufacturer-filter-group input:checked')).map(cb => cb.value);
            const sortBy = document.querySelector('input[name="sort-filter"]:checked')?.value || '';
            return { search, type: types.join(','), manufacturer: manufacturers.join(','), sortBy };
        };
        fetchProducts(getCurrFilters());

        if (searchButton) searchButton.addEventListener('click', () => fetchProducts(getCurrFilters()));
        const applyFiltersBtn = document.getElementById('apply-filters-button');
        if (applyFiltersBtn) applyFiltersBtn.addEventListener('click', () => fetchProducts(getCurrFilters()));
        const resetFiltersBtn = document.getElementById('reset-filters-button');
        if (resetFiltersBtn) resetFiltersBtn.addEventListener('click', () => {
            if (searchInput) searchInput.value = '';
            document.querySelectorAll('#type-filter-group input').forEach(cb => cb.checked = false);
            document.querySelectorAll('#manufacturer-filter-group input').forEach(cb => cb.checked = false);
            const defaultSort = document.querySelector('input[name="sort-filter"][value=""]');
            if (defaultSort) defaultSort.checked = true;
            fetchProducts();
```

```
        });
        document.querySelectorAll('input[name="sort-
filter"]').forEach(radio => radio.addEventListener('change', () =>
fetchProducts(getCurrFilters())));
        document.querySelectorAll('#type-filter-group        input,
#manufacturer-filter-group       input').forEach(checkbox        =>
checkbox.addEventListener('change',              ()              =>
fetchProducts(getCurrFilters())));

        // Функціонал sidebar
        if      (toggleSidebarBtn      &&      filtersSidebar      &&
mainContentWithSidebar) {
            const checkSidebar = () => {
                if (window.innerWidth <= 768) {
                    filtersSidebar.classList.add('collapsed');
                    toggleSidebarBtn.textContent    =    'Показати
Фільтри';
                    mainContentWithSidebar.classList.add('sidebar-
collapsed');
                } else {
                    filtersSidebar.classList.remove('collapsed');
                    toggleSidebarBtn.textContent    =    'Приховати
Фільтри';

mainContentWithSidebar.classList.remove('sidebar-collapsed');
                }
            };
            checkSidebar();
            window.addEventListener('resize', checkSidebar);
            toggleSidebarBtn.addEventListener('click', () => {
                filtersSidebar.classList.toggle('collapsed');
                mainContentWithSidebar.classList.toggle('sidebar-
collapsed');
                toggleSidebarBtn.textContent                        =
filtersSidebar.classList.contains('collapsed') ? 'Показати Фільтри'
: 'Приховати Фільтри';
            });
        }
    }

    // Деталі продукту
    fetchProductDetail();

    // Форми автентифікації
    if       (loginForm)       loginForm.addEventListener('submit',
handleLogin);
    if    (registerForm)    registerForm.addEventListener('submit',
handleRegister);

    // Кошик та оформлення замовлення
    if (cartItemsContainer && checkoutForm) {
        renderCartItems();
        if (deliveryTypeSelect) {
            deliveryTypeSelect.addEventListener('change', () => {
```

```
                if (deliveryTypeSelect.value === 'Відділення Нової
Пошти') {
                    if                        (novaPoshtaDetailsDiv)
novaPoshtaDetailsDiv.style.display = 'block';
                    if                       (homeAddressDetailsDiv)
homeAddressDetailsDiv.style.display = 'none';
                    if                    (novaPoshtaWarehouseInput)
novaPoshtaWarehouseInput.required = true;
                    if (fullAddressInput) fullAddressInput.required
= false;
                    if (fullAddressInput) fullAddressInput.value =
'';
                } else {
                    if                        (novaPoshtaDetailsDiv)
novaPoshtaDetailsDiv.style.display = 'none';
                    if                       (homeAddressDetailsDiv)
homeAddressDetailsDiv.style.display = 'block';
                    if                    (novaPoshtaWarehouseInput)
novaPoshtaWarehouseInput.required = false;
                    if (fullAddressInput) fullAddressInput.required
= true;
                    if                    (novaPoshtaWarehouseInput)
novaPoshtaWarehouseInput.value = '';
                }
            });
            deliveryTypeSelect.dispatchEvent(new  Event('change'));
// Ініціалізація
        }

        checkoutForm.addEventListener('submit', async (e) => {
            e.preventDefault();
            const cart = getCartFromCookie();
            const cartItems = Object.entries(cart).map(([id, qty])
=> ({ id: parseInt(id), quantity: qty }));
            if (cartItems.length === 0) { alert('Кошик порожній!');
return; } // Скорочено

            const   cName   =   document.getElementById('customer-
name').value;
            const   cEmail   =   document.getElementById('customer-
email').value;
            const cPhone = customerPhoneInput.value;
            const   pMethod   =   document.getElementById('payment-
method').value;
            const dType = deliveryTypeSelect.value;
            let sAddress = '';

            if (dType === 'Відділення Нової Пошти') {
                const npWarehouse = novaPoshtaWarehouseInput.value;
                if   (!/^Відділення   №\d+$/.test(npWarehouse))   {
alert('Невірний формат відділення.'); return; } // Скорочено
                sAddress = `Відділення Нової Пошти: ${npWarehouse}`;
            } else sAddress = fullAddressInput.value;
```

```
            if (!cName || !cEmail || !cPhone || !sAddress || !pMethod
|| !dType) { alert('Заповніть усі поля!'); return; } // Скорочено

            try {
                const res = await fetch('/api/orders', { method:
'POST', headers: { 'Content-Type': 'application/json' }, body:
JSON.stringify({ cartItems, customerName: cName, customerEmail:
cEmail, customerPhone: cPhone, shippingAddress: sAddress,
paymentMethod: pMethod, deliveryType: dType }) });
                const result = await res.json();
                if (res.ok) { alert(result.message + ` Номер:
${result.orderId}`); clearCart(); window.location.href = '/my-
orders.html'; } // Скорочено
                else alert(`Помилка: ${result.error ||
'Невідомо'}`); // Скорочено
            } catch (error) {
                console.error('Помилка оформлення замовлення:',
error);
                alert('Помилка оформлення замовлення.'); //
Скорочено
            }
        });
    }

    // Форма зворотного зв'язку
    if (contactForm) {
        contactForm.addEventListener('submit', async (e) => {
            e.preventDefault();
            const name = document.getElementById('name').value;
            const email = document.getElementById('email').value;
            const message =
document.getElementById('message').value;

            try {
                const res = await fetch('/api/contact', { method:
'POST', headers: { 'Content-Type': 'application/json' }, body:
JSON.stringify({ name, email, message }) });
                const result = await res.json();
                if (res.ok) { alert(result.message);
contactForm.reset(); }
                else alert(`Помилка: ${result.error ||
'Невідомо'}`); // Скорочено
            } catch (error) {
                console.error('Помилка відправки повідомлення:',
error);
                alert('Помилка відправки повідомлення.'); //
Скорочено
            }
        });
    }

    // Мої замовлення (для користувача)
    if (userOrdersContainer) fetchUserOrders();

    // Адмін-панель
```

```
    if (document.body.classList.contains('admin-page')) {
        fetchAdminProducts();
        fetchAdminOrders();
        fetchAdminContacts();

        const pEditForm = document.getElementById('product-edit-
form');
        const showAddProductBtn = document.getElementById('show-
add-product-form');
        const cancelEditBtn = document.getElementById('cancel-edit-
btn');
        const pForm = document.getElementById('product-form');

        if (showAddProductBtn)
showAddProductBtn.addEventListener('click', () =>
editProduct(null));
        if (cancelEditBtn) cancelEditBtn.addEventListener('click',
() => { if (pEditForm) pEditForm.style.display = 'none'; });
        if (pForm) pForm.addEventListener('submit', saveProduct);
    }
});
```

server.js

```
const express = require('express');
const db = require('./db'); // Пул підключень
const path = require('path');
const dotenv = require('dotenv');
const bcrypt = require('bcrypt');
const session = require('express-session');
const cookieParser = require('cookie-parser');

dotenv.config(); // Завантажуємо змінні оточення

const app = express();
const port = process.env.PORT || 3000;

// Middleware для статичних файлів, JSON, URL-кодованих даних та
cookie
app.use(express.static(path.join(__dirname, 'public')));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(cookieParser());

// Конфігурація сесій
app.use(session({
    secret: process.env.SESSION_SECRET || 'supersecretkey',
    resave: false,
    saveUninitialized: false,
    cookie: {
        secure: process.env.NODE_ENV === 'production',
        httpOnly: true,
        maxAge: 24 * 60 * 60 * 1000 // 24 години
    }
}));
```

```
// Middleware для перевірки аутентифікації користувача
const requireAuth = (req, res, next) => {
    if (!req.session.userId) {
        return res.status(401).json({ error: 'Необхідна авторизація.' });
    }
    next();
};

// Middleware для перевірки ролі адміністратора
const requireAdmin = (req, res, next) => {
    if (!req.session.isAdmin) {
        return res.status(403).json({ error: 'Доступ заборонено. Потрібні права адміністратора.' });
    }
    next();
};

// API для перевірки аутентифікації
app.get('/api/check-auth', (req, res) => {
    res.json({
        isAuthenticated: !!req.session.userId,
        username: req.session.username || null,
        isAdmin: !!req.session.isAdmin
    });
});
// Реєстрація користувача
app.post('/api/register', (req, res) => {
    const { username, email, password } = req.body;
    if (!username || !email || !password) {
        return res.status(400).json({ error: 'Всі поля обов\'язкові.' });
    }

    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка сервера.' });
        bcrypt.hash(password, 10, (hashErr, hash) => {
            if (hashErr) { connection.release(); return res.status(500).json({ error: 'Помилка хешування пароля.' }); }
            const sql = 'INSERT INTO users (username, email, password_hash) VALUES (?, ?, ?)';
            connection.query(sql, [username, email, hash], (queryErr, result) => {
                connection.release();
                if (queryErr) {
                    if (queryErr.code === 'ER_DUP_ENTRY') {
                        return res.status(409).json({ error: 'Користувач з таким email вже існує.' });
                    }
                    return res.status(500).json({ error: 'Помилка реєстрації.' });
                }
                res.status(201).json({ message: 'Реєстрація успішна!' });
```

```
            });
        });
    });
});
// Вхід користувача
app.post('/api/login', (req, res) => {
    const { email, password } = req.body;
    if (!email || !password) {
        return res.status(400).json({ error: 'Email та пароль
обов\'язкові.' });
    }
    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка
сервера.' });
        const sql = 'SELECT * FROM users WHERE email = ?';
        connection.query(sql, [email], (queryErr, results) => {
            connection.release();
            if (queryErr) return res.status(500).json({ error:
'Помилка входу.' });
            if (results.length === 0) return res.status(401).json({
error: 'Невірний Email або пароль.' });

            const user = results[0];
            bcrypt.compare(password, user.password_hash,
(compareErr, isMatch) => {
                if (compareErr) return res.status(500).json({
error: 'Помилка порівняння паролів.' });
                if (!isMatch) return res.status(401).json({ error:
'Невірний Email або пароль.' });
                req.session.userId = user.id;
                req.session.username = user.username;
                req.session.isAdmin = user.is_admin;
                res.status(200).json({ message: 'Вхід успішний!',
username: user.username, isAdmin: user.is_admin });
            });
        });
    });
});
// Вихід користувача
app.post('/api/logout', (req, res) => {
    req.session.destroy(err => {
        if (err) return res.status(500).json({ error: 'Помилка
виходу.' });
        res.clearCookie('connect.sid'); // Очистити cookie сесії
        res.status(200).json({ message: 'Вихід успішний.' });
    });
});
// Отримання всіх продуктів (для клієнтської частини та адмінки)
app.get('/api/products', (req, res) => {
    const { search, type, manufacturer, sortBy } = req.query;
    let sql = 'SELECT * FROM products WHERE 1=1';
    const params = [];
    if (search) {
        sql += ' AND (name LIKE ? OR description LIKE ?)';
        params.push(`%${search}%`, `%${search}%`);
```

```
    }
    if (type) {
        const types = type.split(',').map(t => t.trim());
        sql += ` AND type IN (${types.map(() => '?').join(',')})`;
        params.push(...types);
    }
    if (manufacturer) {
        const manufacturers = manufacturer.split(',').map(m =>
m.trim());
        sql += ` AND manufacturer IN (${manufacturers.map(() =>
'?').join(',')})`;
        params.push(...manufacturers);
    }
    if (sortBy === 'price_asc') {
        sql += ' ORDER BY price ASC';
    } else if (sortBy === 'price_desc') {
        sql += ' ORDER BY price DESC';
    } else {
        sql += ' ORDER BY name ASC';
    }
    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка
сервера.' });
        connection.query(sql, params, (queryErr, results) => {
            connection.release();
            if (queryErr) return res.status(500).json({ error:
'Помилка отримання продуктів.' });
            res.json(results);
        });
    });
});
// Отримання популярних продуктів (для головної сторінки)
app.get('/api/popular-products', (req, res) => {
    const sql = 'SELECT p.*, SUM(oi.quantity) AS total_sold FROM
products p JOIN order_items oi ON p.id = oi.product_id GROUP BY p.id
ORDER BY total_sold DESC LIMIT 8';
    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка
сервера.' });
        connection.query(sql, (queryErr, results) => {
            connection.release();
            if (queryErr) return res.status(500).json({ error:
'Помилка отримання популярних продуктів.' });
            res.json(results);
        });
    });
});
// Отримання одного продукту за ID
app.get('/api/products/:id', (req, res) => {
    const productId = req.params.id;
    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка
сервера.' });
        const sql = 'SELECT * FROM products WHERE id = ?';
        connection.query(sql, [productId], (queryErr, results) => {
```

```
            connection.release();
            if (queryErr) return res.status(500).json({ error:
'Помилка отримання продукту.' });
            if (results.length === 0) return res.status(404).json({
error: 'Продукт не знайдено.' });
            res.json(results[0]);
        });
    });
});
// Отримання товарів кошика за їх ID
app.post('/api/cart/items', (req, res) => {
    const { productIds } = req.body;
    if (!productIds || !Array.isArray(productIds) ||
productIds.length === 0) {
        return res.status(400).json({ error: 'Недійсні ID
продуктів.' });
    }
    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка
сервера.' });
        const placeholders = productIds.map(() => '?').join(',');
        const sql = `SELECT id, name, price, image_url FROM products
WHERE id IN (${placeholders})`;
        connection.query(sql, productIds, (queryErr, results) => {
            connection.release();
            if (queryErr) return res.status(500).json({ error:
'Помилка отримання даних продуктів кошика.' });
            res.json(results);
        });
    });
});
// Додавання нового продукту (тільки для адміна)
app.post('/api/admin/products', requireAdmin, (req, res) => {
    const { name, description, price, type, manufacturer, image_url,
weight_grams } = req.body;
    if (!name || !price || !type || !manufacturer || !image_url) {
        return res.status(400).json({ error: 'Основні поля
обов\'язкові.' });
    }
    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка
сервера.' });
        const sql = 'INSERT INTO products (name, description, price,
type, manufacturer, image_url, weight_grams) VALUES (?, ?, ?, ?, ?,
?, ?)';
        connection.query(sql, [name, description, price, type,
manufacturer, image_url, weight_grams], (queryErr, result) => {
            connection.release();
            if (queryErr) return res.status(500).json({ error:
'Помилка додавання продукту.' });
            res.status(201).json({ message: 'Продукт успішно
додано!', productId: result.insertId });
        });
    });
});
```

```
// Оновлення продукту (тільки для адміна)
app.put('/api/admin/products/:id', requireAdmin, (req, res) => {
    const productId = req.params.id;
    const { name, description, price, type, manufacturer, image_url,
weight_grams } = req.body;
    if (!name || !price || !type || !manufacturer || !image_url) {
        return res.status(400).json({ error: 'Основні поля
обов\'язкові.' });
    }
    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка
сервера.' });
        const sql = 'UPDATE products SET name=?, description=?,
price=?, type=?, manufacturer=?, image_url=?, weight_grams=? WHERE
id=?';
        connection.query(sql, [name, description, price, type,
manufacturer, image_url, weight_grams, productId], (queryErr,
result) => {
            connection.release();
            if (queryErr) return res.status(500).json({ error:
'Помилка оновлення продукту.' });
            if (result.affectedRows === 0) return
res.status(404).json({ error: 'Продукт не знайдено.' });
            res.status(200).json({ message: 'Продукт успішно
оновлено!' });
        });
    });
});
// Видалення продукту (тільки для адміна)
app.delete('/api/admin/products/:id', requireAdmin, (req, res) => {
    const productId = req.params.id;
    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка
сервера.' });
        const sql = 'DELETE FROM products WHERE id = ?';
        connection.query(sql, [productId], (queryErr, result) => {
            connection.release();
            if (queryErr) return res.status(500).json({ error:
'Помилка видалення продукту.' });
            if (result.affectedRows === 0) return
res.status(404).json({ error: 'Продукт не знайдено.' });
            res.status(200).json({ message: 'Продукт успішно
видалено!' });
        });
    });
});

// Оформлення замовлення
app.post('/api/orders', async (req, res) => {
    const { cartItems, customerName, customerEmail, customerPhone,
shippingAddress, paymentMethod, deliveryType } = req.body;
    if (!cartItems || cartItems.length === 0 || !customerName ||
!customerEmail || !shippingAddress || !paymentMethod ||
!deliveryType) {
```

```
        return res.status(400).json({ error: 'Всі обов\'язкові поля
замовлення мають бути заповнені.' });
    }

    let connection;
    try {
        connection = await new Promise((resolve, reject) => {
            db.getConnection((err, conn) => {
                if (err) return reject(err);
                resolve(conn);
            });
        });

        await connection.beginTransaction();

        let totalAmount = 0;
        const productsData = {};

        // Отримання цін продуктів
        const productIds = cartItems.map(item => item.id);
        if (productIds.length > 0) {
            const [products] = await connection.execute(
                `SELECT id, price FROM products WHERE id IN
(${productIds.map(() => '?').join(',')})`,
                productIds
            );
            products.forEach(p => { productsData[p.id] = p; });
        }

        // Обчислення загальної суми та підготовка товарів
        const orderItems = cartItems.map(item => {
            const product = productsData[item.id];
            if (!product) throw new Error(`Продукт з ID ${item.id}
не знайдено.`);
            const itemPrice = product.price;
            totalAmount += itemPrice * item.quantity;
            return [item.id, item.quantity, itemPrice];
        });

        // Додавання замовлення в таблицю orders
        const [orderResult] = await connection.execute(
            'INSERT INTO orders (user_id, customer_name,
customer_email, customer_phone, total_amount, shipping_address,
payment_method, delivery_type) VALUES (?, ?, ?, ?, ?, ?, ?, ?)',
            [req.session.userId || null, customerName,
customerEmail, customerPhone, totalAmount, shippingAddress,
paymentMethod, deliveryType]
        );
        const orderId = orderResult.insertId;
        // Додавання товарів замовлення в таблицю order_items
        for (const item of orderItems) {
            await connection.execute(
                'INSERT INTO order_items (order_id, product_id,
quantity, item_price) VALUES (?, ?, ?, ?)',
                [orderId, item[0], item[1], item[2]]
```

```
            );
        }
        await connection.commit();
        res.status(201).json({ message: 'Замовлення успішно
оформлено!', orderId: orderId });

    } catch (error) {
        if (connection) connection.rollback(() =>
connection.release());
        console.error('Помилка оформлення замовлення:', error);
        res.status(500).json({ error: error.message || 'Помилка
сервера при оформленні замовлення.' });
    } finally {
        if (connection) connection.release();
    }
});

// Отримання всіх замовлень (тільки для адміна)
app.get('/api/admin/orders', requireAdmin, (req, res) => {
    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка
сервера.' });
        const sql = `
            SELECT o.*, u.username AS user_username,
                   GROUP_CONCAT(CONCAT(p.name, ' (', oi.quantity, '
шт., ', oi.item_price, ' грн) - ', p.image_url) SEPARATOR '|||') AS
items_data
            FROM orders o
            LEFT JOIN users u ON o.user_id = u.id
            LEFT JOIN order_items oi ON o.id = oi.order_id
            LEFT JOIN products p ON oi.product_id = p.id
            GROUP BY o.id
            ORDER BY o.created_at DESC
        `;
        connection.query(sql, (queryErr, results) => {
            connection.release();
            if (queryErr) return res.status(500).json({ error:
'Помилка отримання замовлень.' });
            const ordersWithParsedItems = results.map(order => {
                const items = order.items_data ?
order.items_data.split('|||').map(itemStr => {
                    const parts = itemStr.match(/(.*) \((\d+) шт\.,
([\d.]+) грн\) - (.*)/);
                    return parts ? {
                        product_name: parts[1].trim(),
                        quantity: parseInt(parts[2]),
                        item_price: parseFloat(parts[3]),
                        image_url: parts[4].trim()
                    } : null;
                }).filter(Boolean) : [];
                return { ...order, items: items };
            });
            res.json(ordersWithParsedItems);
        });
    });
```

```
});

// Оновлення статусу замовлення (тільки для адміна)
app.put('/api/admin/orders/:id/status', requireAdmin, (req, res) =>
{
    const orderId = req.params.id;
    const { status } = req.body;
    if (!status) return res.status(400).json({ error: 'Статус
замовлення обов\'язковий.' });

    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка
сервера.' });
        const sql = 'UPDATE orders SET status = ? WHERE id = ?';
        connection.query(sql, [status, orderId], (queryErr, result)
=> {
            connection.release();
            if (queryErr) return res.status(500).json({ error:
'Помилка оновлення статусу замовлення.' });
            if (result.affectedRows === 0) return
res.status(404).json({ error: 'Замовлення не знайдено.' });
            res.status(200).json({ message: 'Статус замовлення
успішно оновлено!' });
        });
    });
});

// Отримання замовлень поточного користувача
app.get('/api/my-orders', requireAuth, (req, res) => {
    const userId = req.session.userId;
    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка
сервера.' });
        const sql = `
            SELECT o.*, GROUP_CONCAT(CONCAT(p.name, ' (',
oi.quantity, ' шт., ', oi.item_price, ' грн) - ', p.image_url)
SEPARATOR '|||') AS items_data
            FROM orders o
            LEFT JOIN order_items oi ON o.id = oi.order_id
            LEFT JOIN products p ON oi.product_id = p.id
            WHERE o.user_id = ?
            GROUP BY o.id
            ORDER BY o.created_at DESC
        `;
        connection.query(sql, [userId], (queryErr, results) => {
            connection.release();
            if (queryErr) return res.status(500).json({ error:
'Помилка отримання ваших замовлень.' });

            const ordersWithParsedItems = results.map(order => {
                const items = order.items_data ?
order.items_data.split('|||').map(itemStr => {
                    const parts = itemStr.match(/(.*) \((\d+) шт\.,
([\d.]+) грн\) - (.*)/);
                    return parts ? {
```

```
                        product_name: parts[1].trim(),
                        quantity: parseInt(parts[2]),
                        item_price: parseFloat(parts[3]),
                        image_url: parts[4].trim()
                    } : null;
                }).filter(Boolean) : [];
                return { ...order, items: items };
            });
            res.json(ordersWithParsedItems);
        });
    });
});

// Відправлення повідомлення зворотного зв'язку
app.post('/api/contact', (req, res) => {
    const { name, email, message } = req.body;
    if (!name || !email || !message) {
        return res.status(400).json({ error: 'Всі поля форми зворотного зв\'язку обов\'язкові.' });
    }

    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка сервера.' });
        const sql = 'INSERT INTO contacts (name, email, message) VALUES (?, ?, ?)';
        connection.query(sql, [name, email, message], (queryErr, result) => {
            connection.release();
            if (queryErr) return res.status(500).json({ error: 'Помилка при збереженні повідомлення.' });
            res.status(201).json({ message: 'Повідомлення успішно відправлено!' });
        });
    });
});

// Отримання всіх повідомлень зворотного зв'язку (тільки для адміна)
app.get('/api/admin/contacts', requireAdmin, (req, res) => {
    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка сервера.' });
        const sql = 'SELECT * FROM contacts ORDER BY created_at DESC';
        connection.query(sql, (queryErr, results) => {
            connection.release();
            if (queryErr) return res.status(500).json({ error: 'Помилка отримання повідомлень.' });
            res.json(results);
        });
    });
});
// Оновлення статусу повідомлення зворотного зв'язку (тільки для адміна)
```

```
app.put('/api/admin/contacts/:id/status', requireAdmin, (req, res) => {
    const contactId = req.params.id;
    const { status } = req.body;
    if (!status) return res.status(400).json({ error: 'Статус повідомлення обов\'язковий.' });
    db.getConnection((err, connection) => {
        if (err) return res.status(500).json({ error: 'Помилка сервера.' });
        const sql = 'UPDATE contacts SET status = ? WHERE id = ?';
        connection.query(sql, [status, contactId], (queryErr, result) => {
            connection.release();
            if (queryErr) return res.status(500).json({ error: 'Помилка оновлення статусу повідомлення.' });
            if (result.affectedRows === 0) return res.status(404).json({ error: 'Повідомлення не знайдено.' });
            res.status(200).json({ message: 'Статус повідомлення успішно оновлено!' });
        });
    });
});
// Обробка неіснуючих маршрутів
app.use((req, res, next) => {
    res.status(404).send('Сторінку не знайдено.');
});
// Глобальний обробник помилок
app.use((err, req, res, next) => {
    console.error(err.stack);
    res.status(500).send('Щось пішло не так на сервері!');
});
// Запуск сервера
app.listen(port, () => {
    console.log(`Сервер працює на http://localhost:${port}`);
});
```

db.js

```
// db.js
const mysql = require('mysql');
const dotenv = require('dotenv');

dotenv.config(); // Завантажуємо змінні оточення

// ЗМІНЕНО: Використовуємо пул підключень замість одного підключення
const pool = mysql.createPool({
    host: process.env.DB_HOST,
    user: process.env.DB_USER,
    password: process.env.DB_PASSWORD,
    database: process.env.DB_NAME,
    connectionLimit: 10, // Максимальна кількість одночасних підключень у пулі
    waitForConnections: true, // Чи чекати, якщо всі підключення зайняті
```

```
    queueLimit: 0 // Немає обмеження для черги (0 означає без
обмежень)
});
// Перевірка з'єднання при запуску пулу (не обов'язково, але корисно
для діагностики)
pool.getConnection((err, connection) => {
    if (err) {
        console.error('Помилка підключення до пулу бази даних:',
err.message);
        // Завершити процес, якщо не вдалося підключитися до БД
        process.exit(1);
    }
    console.log('Пул підключень до бази даних MySQL успішно
ініціалізовано.');
    connection.release(); // Важливо: повернути підключення в пул
одразу після перевірки
});
// Експортуємо об'єкт пулу, щоб його можна було використовувати в
інших файлах
module.exports = pool;
```

admin.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta  name="viewport"  content="width=device-width,  initial-
scale=1.0">
    <title>IronLab - Адмін</title>
    <link rel="stylesheet" href="css/style.css">
    <link        rel="icon"        href="/images/image_43af6e.png"
type="image/png">
</head>

<body class="admin-page">
    <header>
        <h1><a href="/">IronLab</a></h1>
        <nav>
            <ul>
                <li><a href="/">Головна</a></li>
                <li><a href="/products.html">Продукти</a></li>
                <li><a href="/about.html">Про нас</a></li>
                <li><a href="/contact.html">Контакти</a></li>
                <li><a  href="/cart.html">Кошик  (<span  id="cart-
count">0</span>)</a></li>
                <li                          id="my-orders-nav-link"
style="display:none;"><a                href="/my-orders.html">Мої
Замовлення</a></li>
                <li    id="admin-link"    style="display:none;"><a
href="/admin.html">Адмін-панель</a></li>
                <li                                id="auth-link"><a
href="/register.html">Реєстрація</a></li>
            </ul>
        </nav>
```

```
    </header>

    <main>
        <section id="admin-panel-content">
            <h2>Адміністративна Панель</h2>

            <div class="admin-section">
                <h3>Управління Продуктами</h3>
                <div id="product-edit-form">
                    <h4    id="product-form-title">Додати    новий
продукт</h4>
                    <form id="product-form">
                        <input type="hidden" id="product-id">
                        <label for="product-name">Назва:</label>
                        <input    type="text"    id="product-name"
required>

                        <label                          for="product-
description">Опис:</label>
                        <textarea          id="product-description"
rows="4"></textarea>

                        <label for="product-price">Ціна:</label>
                        <input   type="number"   id="product-price"
step="0.01" min="0" required>

                        <label for="product-type">Тип:</label>
                        <input    type="text"    id="product-type"
required>

                        <label                          for="product-
manufacturer">Виробник:</label>
                        <input       type="text"       id="product-
manufacturer" required>

                        <label          for="product-image-url">URL
Зображення:</label>
                        <input  type="text"  id="product-image-url"
required>

                        <!--  ПОЛЕ  ДЛЯ  ВАГИ  -  ВИПРАВЛЕНО  ID  НА
"product-weight-grams" -->
                        <label      for="product-weight-grams">Вага
(грами):</label>
                        <input   type="number"   id="product-weight-
grams" step="0.01" min="0">

                        <button   type="submit"   id="save-product-
btn">Зберегти</button>
                        <button type="button" id="cancel-edit-btn"
class="cancel-edit-btn">Скасувати</button>
                    </form>
                </div>

                <div class="product-admin-controls">
```

```
                    <button id="show-add-product-form" class="add-
product-btn">Додати новий продукт</button>
                </div>
                <div id="products-list">
                    <p>Завантаження продуктів...</p>
                </div>
            </div>
            <div class="admin-section">
                <h3>Управління Замовленнями</h3>
                <div id="orders-list">
                    <p>Завантаження замовлень...</p>
                </div>
            </div>
            <div class="admin-section">
                <h3>Повідомлення Зворотного Зв'язку</h3>
                <div id="contacts-list">
                    <p>Завантаження повідомлень...</p>
                </div>
            </div>
        </section>
    </main>
    <footer>
        <p>&copy; 2025 IronLab. Усі права захищені.</p>
    </footer>
    <script src="js/script.js"></script>
</body>
</html>
```

cart.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta  name="viewport"  content="width=device-width,  initial-
scale=1.0">
    <title>IronLab - Кошик</title>
    <link rel="stylesheet" href="css/style.css">
    <link rel="icon" href="/images/x-icon.png" type="image/png">
</head>
<body>
    <header>
        <h1><a href="/">IronLab</a></h1>
        <nav>
            <ul>
                <li><a href="/">Головна</a></li>
                <li><a href="/products.html">Продукти</a></li>
                <li><a href="/about.html">Про нас</a></li>
                <li><a href="/contact.html">Контакти</a></li>
                <li><a  href="/cart.html">Кошик  (<span  id="cart-
count">0</span>)</a></li>
                <li                          id="my-orders-nav-link"
style="display:none;"><a                href="/my-orders.html">Мої
Замовлення</a></li>
```

```
                <li    id="admin-link"    style="display:none;"><a
href="/admin.html">Адмін-панель</a></li>
                <li                                id="auth-link"><a
href="/register.html">Реєстрація</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <h2>Ваш кошик</h2>
        <section id="cart-content">
            <div id="cart-items">
                <p>Завантаження кошика...</p>
            </div>
            <div class="cart-summary">
                <p>Загальна   сума:   <span   id="cart-total">0.00
грн</span></p>
                <!--   Додано   клас   checkout-form-container   для
застосування стилів -->
                <div  class="checkout-form-section  checkout-form-
container">
                    <h3>Дані для оформлення замовлення</h3>
                    <form id="checkout-form">
                        <label             for="customer-name">Ваше
ім'я:</label>
                        <input    type="text"    id="customer-name"
placeholder="Введіть ваше ім'я та прізвище" required>
                        <label             for="customer-email">Ваш
Email:</label>
                        <input   type="email"   id="customer-email"
placeholder="Введіть ваш email" required>
                        <label    for="customer-phone">Ваш    номер
телефону:</label>
                        <input    type="tel"    id="customer-phone"
placeholder="Наприклад,  +380XXXXXXXXX"  required  pattern="^\+?[0-
9]{10,15}$" title="Будь ласка, введіть дійсний номер телефону (10-
15 цифр, може починатися з +)">
                        <label              for="delivery-type">Тип
доставки:</label>
                        <select id="delivery-type" required>
                            <option         value="Додому">Доставка
додому</option>
                            <option     value="Відділення     Нової
Пошти">Доставка у відділення Нової Пошти</option>
                        </select>
                        <div               id="nova-poshta-details"
style="display: none;">
                            <label                for="nova-poshta-
warehouse">Повна адреса та номер відділення Нової Пошти:</label>
                            <input   type="text"   id="nova-poshta-
warehouse" placeholder="Наприклад,вул, місто, обл, відділення №"
title="Введіть 'вул, місто, обл, відділення №'">
                        </div>
                        <div id="home-address-details">
                            <label  for="full-address">Повна адреса
доставки (вул, буд, кв, місто, обл, індекс):</label>
```

```
                            <input  type="text"  id="full-address"
placeholder="Наприклад, вул. Шевченка, 15, кв. 3, м. Київ, Київська
обл., 01001" required>
                        </div>
                        <label           for="payment-method">Метод
оплати:</label>
                        <select id="payment-method" required>
                            <option     value="">Оберіть     метод
оплати</option>
                            <option   value="Карта   онлайн">Оплата
карткою онлайн</option>
                            <option  value="При  отриманні">Готівка
при отриманні</option>
                            <option              value="Банківський
переказ">Банківський переказ</option>
                        </select>
                        <button     type="submit"     id="checkout-
button">Оформити замовлення</button>
                    </form>
                </div>
            </div>
        </section>
    </main
    <footer>
        <p>&copy; 2025 IronLab. Усі права захищені.</p>
    </footer>
    <script src="js/script.js"></script>
</body>
</html>
```

login.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta  name="viewport"  content="width=device-width,  initial-
scale=1.0">
    <title>IronLab - Вхід</title>
    <link rel="stylesheet" href="css/style.css">
    <link        rel="icon"        href="/images/image_43af6e.png"
type="image/png">
</head>
<body>
    <header>
        <h1><a href="/">IronLab</a></h1>
        <nav>
            <ul>
                <li><a href="/">Головна</a></li>
                <li><a href="/products.html">Продукти</a></li>
                <li><a href="/about.html">Про нас</a></li>
                <li><a href="/contact.html">Контакти</a></li>
                <li><a  href="/cart.html">Кошик  (<span  id="cart-
count">0</span>)</a></li>
```

```
                <li                              id="my-orders-nav-link"
style="display:none;"><a                    href="/my-orders.html">Мої
Замовлення</a></li>
                <li    id="admin-link"    style="display:none;"><a
href="/admin.html">Адмін-панель</a></li>
                <li                                    id="auth-link"><a
href="/register.html">Реєстрація</a></li>
            </ul>
        </nav>
    </header>

    <main>
        <!-- Додано клас auth-form-container для застосування стилів
-->
        <section id="login-content" class="auth-form-container">
            <h2>Вхід</h2>
            <form id="login-form">
                <label for="login-email">Електронна пошта:</label>
                <input  type="email"  id="login-email"  name="email"
required>

                <label for="login-password">Пароль:</label>
                <input     type="password"     id="login-password"
name="password" required>

                <button type="submit">Увійти</button>
            </form>
            <p>Ще     не     маєте     облікового     запису?     <a
href="/register.html">Зареєструватися</a></p>
        </section>
    </main>

    <footer>
        <p>&copy; 2025 IronLab. Усі права захищені.</p>
    </footer>
    <script src="js/script.js"></script>
</body>
</html>
```

products.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta  name="viewport"  content="width=device-width,  initial-
scale=1.0">
    <title>IronLab - Продукти</title>
    <link rel="stylesheet" href="css/style.css">
    <link        rel="icon"        href="/images/image_43af6e.png"
type="image/png">
</head>
<body class="products-page">
    <!-- ДОДАНО КЛАС products-page -->
    <header>
```

```
        <h1><a href="/">IronLab</a></h1>
        <nav>
            <ul>
                <li><a href="/">Головна</a></li>
                <li><a href="/products.html">Продукти</a></li>
                <li><a href="/about.html">Про нас</a></li>
                <li><a href="/contact.html">Контакти</a></li>
                <li><a href="/cart.html">Кошик (<span id="cart-
count">0</span>)</a></li>
                <li id="my-orders-nav-link" 
style="display:none;"><a href="/my-orders.html">Мої 
Замовлення</a></li>
                <li id="admin-link" style="display:none;"><a 
href="/admin.html">Адмін-панель</a></li>
                <li id="auth-link"><a 
href="/register.html">Реєстрація</a></li>
            </ul>
        </nav>
    </header>

    <main class="has-sidebar">
        <aside class="filters-sidebar" id="filters-sidebar">
            <div class="filter-section">
                <h3>Пошук за назвою</h3>
                <div class="search-bar">
                    <input type="text" id="search-input" 
placeholder="Пошук продуктів...">
                    <button id="search-button">Знайти</button>
                </div>
            </div>

            <div class="filter-section">
                <h3>Фільтрувати за типом</h3>
                <div class="filter-group" id="type-filter-group">
                    <label><input type="checkbox" value="Протеїн">
Протеїн</label>
                    <label><input type="checkbox" value="Креатин">
Креатин</label>
                    <label><input type="checkbox" value="BCAA">
BCAA</label>
                </div>
            </div>

            <div class="filter-section">
                <h3>Фільтрувати за виробником</h3>
                <div class="filter-group" id="manufacturer-filter-
group">
                    <label><input type="checkbox" value="Optimum 
Nutrition"> Optimum Nutrition</label>
                    <label><input type="checkbox" value="Kevin 
Levrone"> Kevin Levrone</label>
                    <label><input type="checkbox" value="Cellucor">
Cellucor</label>
                    <label><input type="checkbox" value="Universal 
Nutrition"> Universal Nutrition</label>
```

```
                    <label><input                    type="checkbox"
value="MusclePharm"> MusclePharm</label>
                </div>
            </div>

            <div class="filter-section">
                <h3>Сортування</h3>
                <div class="sort-group">
                    <label><input  type="radio"  name="sort-filter"
value="" checked> Без сортування</label>
                    <label><input  type="radio"  name="sort-filter"
value="price_asc"> Ціна: від низької до високої</label>
                    <label><input  type="radio"  name="sort-filter"
value="price_desc"> Ціна: від високої до низької</label>
                    <label><input  type="radio"  name="sort-filter"
value="name_asc"> Назва: від А до Я</label>
                    <label><input  type="radio"  name="sort-filter"
value="name_desc"> Назва: від Я до А</label>
                    <label><input  type="radio"  name="sort-filter"
value="weight_asc"> Вага: від малої до великої</label>
                    <label><input  type="radio"  name="sort-filter"
value="weight_desc"> Вага: від великої до малої</label>
                </div>
            </div>

            <div class="filter-actions">
                <button       id="apply-filters-button">Застосувати
фільтри</button>
                <button id="reset-filters-button">Скинути</button>
            </div>
        </aside>

        <section id="product-list-section">
            <button  id="toggle-sidebar-btn"  class="toggle-sidebar-
btn">Приховати Фільтри</button> <!-- Кнопка для перемикання сайдбару
-->
            <div id="product-list-container">
                <p>Завантаження продуктів...</p>
            </div>
        </section>
    </main>

    <footer>
        <p>&copy; 2025 IronLab. Усі права захищені.</p>
    </footer>

    <script src="js/script.js"></script>
</body>
</html>
```

style.css

```
/* public/css/style.css */

/* Завантажуємо шрифт Inter з Google Fonts */
```

```
@import
url('https://fonts.googleapis.com/css2?family=Inter:wght@300;400;6
00;700&display=swap');

body {
    font-family: 'Inter', sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4; /* Світло-сірий фон */
    color: #333;
}

/* Кругла рамка для favicon */
link[rel="icon"] {
    border-radius: 50%;
}

header {
    background-color: #333; /* Темно-сірий */
    color: white;
    padding: 1em 20px;
    display: flex;
    justify-content: space-between;
    align-items: center;
    flex-wrap: wrap;
}

    header h1 {
        margin: 0;
        order: 0;
    }

    header nav ul {
        list-style: none;
        padding: 0;
        margin: 0;
        display: flex;
        gap: 20px;
        order: 1; /* Змінюємо порядок для мобільної адаптації */
        flex-wrap: wrap;
        justify-content: center;
    }

    header nav a {
        color: white;
        text-decoration: none;
        font-weight: 600;
        padding: 5px 10px;
        transition: background-color 0.3s ease;
    }

    header nav a:hover,
    header nav a.active {
        background-color: #555;
        border-radius: 5px;
```

```
    }

    #cart-link {
        position: relative;
    }

    #cart-count {
        background-color: #ff5722; /* Яскравий акцент */
        color: white;
        border-radius: 50%;
        padding: 2px 7px;
        font-size: 0.8em;
        position: absolute;
        top: -8px;
        right: -8px;
    }

/* Основний вміст сторінки */
main {
    padding: 20px;
    max-width: 1200px;
    margin: 20px auto;
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

main h2, main h3 {
    color: #333;
    border-bottom: 2px solid #eee;
    padding-bottom: 10px;
    margin-bottom: 20px;
}

/* Форми */
form {
    display: flex;
    flex-direction: column;
    gap: 15px;
    max-width: 500px;
    margin: 20px auto;
    padding: 20px;
    border: 1px solid #ddd;
    border-radius: 8px;
    background-color: #f9f9f9;
}

form label {
    font-weight: 600;
    margin-bottom: 5px;
}

form input[type="text"],
form input[type="email"],
form input[type="password"],
```

```
form input[type="number"],
form textarea,
form select {
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
    font-size: 1em;
    width: 100%;
    box-sizing: border-box; /* Враховуємо padding у загальній ширині
*/
}

form button,
.button {
    background-color: #4CAF50; /* Зелений */
    color: white;
    padding: 10px 15px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 1em;
    transition: background-color 0.3s ease;
    width: auto;
    align-self: flex-start; /* Вирівнювання кнопки */
}

form button:hover,
.button:hover {
    background-color: #45a049;
}

.error-message {
    color: red;
    font-size: 0.9em;
    margin-top: -10px;
    margin-bottom: 10px;
}

/* Сторінка продуктів */
#product-list-container {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
    gap: 20px;
    margin-top: 20px;
}

.product-card {
    background-color: white;
    border: 1px solid #ddd;
    border-radius: 8px;
    overflow: hidden;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
    display: flex;
    flex-direction: column;
    text-align: center;
```

```
    transition: transform 0.2s ease;
}

.product-card:hover {
    transform: translateY(-5px);
}

.product-card-link {
    text-decoration: none;
    color: inherit;
    display: flex;
    flex-direction: column;
    flex-grow: 1;
}

.product-card img {
    width: 100%;
    height: 200px;
    object-fit: cover;
    border-bottom: 1px solid #eee;
}

.product-card h3 {
    font-size: 1.2em;
    margin: 15px 10px 5px;
    color: #333;
    border-bottom: none;
    padding-bottom: 0;
}

.product-card .product-details {
    padding: 0 10px;
    text-align: left;
    margin-bottom: 10px;
    flex-grow: 1;
}

.product-card .product-details ul {
    list-style: none;
    padding: 0;
    margin: 0;
    font-size: 0.9em;
    color: #666;
}

.product-card .product-details li {
    margin-bottom: 3px;
}

.product-card .price {
    font-size: 1.3em;
    font-weight: 700;
    color: #e67e22; /* Помаранчевий */
    margin-bottom: 15px;
}
```

```
.add-to-cart-btn {
    background-color: #007bff; /* Синій */
    color: white;
    padding: 10px;
    border: none;
    cursor: pointer;
    transition: background-color 0.3s ease;
    width: 100%;
    border-radius: 0 0 8px 8px; /* Заокруглюємо тільки низ кнопки
*/
}

.add-to-cart-btn:hover {
    background-color: #0056b3;
}

/* Деталі продукту */
#product-detail {
    display: flex;
    justify-content: center;
    padding: 20px;
}

.product-detail-card {
    display: flex;
    flex-wrap: wrap; /* Дозволяє переносити елементи на новий рядок
*/
    gap: 30px;
    background-color: white;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    max-width: 900px;
    padding: 30px;
}

.product-detail-image {
    width: 100%;
    max-width: 400px;
    height: auto;
    object-fit: contain; /* Зберігаємо пропорції */
    border-radius: 8px;
}

.product-info {
    flex: 1;
    min-width: 300px; /* Мінімальна ширина для інформаційного блоку
*/
}

.product-info h2 {
    font-size: 2em;
    margin-top: 0;
    color: #333;
    border-bottom: 2px solid #eee;
```

```
    padding-bottom: 10px;
}

.product-info .product-description {
    font-size: 1.1em;
    line-height: 1.6;
    color: #555;
    margin-bottom: 20px;
}

.product-info .product-price {
    font-size: 1.8em;
    font-weight: 700;
    color: #e67e22;
    margin-bottom: 15px;
}

.product-info p {
    font-size: 1em;
    color: #666;
    margin-bottom: 8px;
}

.product-info .add-to-cart-btn {
    width: auto; /* Кнопка не займає всю ширину */
    padding: 12px 25px;
    font-size: 1.1em;
    margin-top: 20px;
    border-radius: 5px;
}

/* Сторінка кошика */
#cart-items {
    margin-top: 20px;
}

.cart-item {
    display: flex;
    align-items: center;
    gap: 20px;
    margin-bottom: 15px;
    padding: 15px;
    border: 1px solid #eee;
    border-radius: 8px;
    background-color: #fcfcfc;
}

.cart-item img {
    width: 80px;
    height: 80px;
    object-fit: cover;
    border-radius: 4px;
}

.cart-item .item-details {
```

```
    flex-grow: 1;
}
.cart-item h4 {
    margin: 0 0 5px;
    font-size: 1.1em;
}
.cart-item p {
    margin: 0 0 5px;
    color: #555;
}
.cart-item .quantity-controls {
    display: flex;
    align-items: center;
    gap: 10px;
    margin-bottom: 10px;
}
.cart-item .quantity-controls button {
    background-color: #007bff;
    color: white;
    border: none;
    border-radius: 4px;
    padding: 5px 10px;
    cursor: pointer;
    font-size: 1em;
}
.cart-item .quantity-controls button:hover {
    background-color: #0056b3;
}
.cart-item .remove-from-cart-btn {
    background-color: #dc3545; /* Червоний */
    color: white;
    border: none;
    border-radius: 4px;
    padding: 8px 12px;
    cursor: pointer;
    font-size: 0.9em;
    transition: background-color 0.3s ease;
}
.cart-item .remove-from-cart-btn:hover {
    background-color: #c82333;
}
.cart-summary {
    text-align: right;
    margin-top: 20px;
    font-size: 1.2em;
    font-weight: 700;
}
.cart-summary span {
    color: #e67e22;
}
.checkout-form-container {
    margin-top: 30px;
    padding-top: 20px;
    border-top: 1px solid #eee;
}
```

```
#delivery-details div {
    display: none; /* За замовчуванням приховуємо */
    margin-top: 10px;
}
#delivery-details input {
    margin-top: 5px;
}
/* Сторінка "Мої Замовлення" */
.order-card {
    background-color: #f9f9f9;
    border: 1px solid #ddd;
    border-radius: 8px;
    padding: 20px;
    margin-bottom: 20px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.05);
}
.order-card h3 {
    margin-top: 0;
    color: #333;
    border-bottom: 1px solid #eee;
    padding-bottom: 10px;
}

.order-card p {
    margin-bottom: 8px;
    color: #555;
}
.order-card ul {
    list-style: none;
    padding: 0;
    margin: 10px 0 0;
}
.order-card li {
    margin-bottom: 5px;
    color: #666;
}
/* Статуси замовлень */
.order-status {
    font-weight: 700;
    padding: 3px 8px;
    border-radius: 4px;
    color: white;
}
.order-status.pending { background-color: #ffc107; } /* Жовтий */
.order-status.processing { background-color: #17a2b8; } /* Блакитний
*/
.order-status.shipped { background-color: #007bff; } /* Синій */
.order-status.completed { background-color: #28a745; } /* Зелений */
.order-status.cancelled { background-color: #dc3545; } /* Червоний
*/


/* Адмін-панель */
.admin-section {
    margin-bottom: 40px;
```

```
    padding: 20px;
    border: 1px solid #eee;
    border-radius: 8px;
    background-color: #f9f9f9;
}
.admin-section h2 {
    margin-top: 0;
}
.products-admin-table,
.orders-table,
.contacts-table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}
.products-admin-table th, .products-admin-table td,
.orders-table th, .orders-table td,
.contacts-table th, .contacts-table td {
    border: 1px solid #ddd;
    padding: 8px;
    text-align: left;
    vertical-align: top;
}
.products-admin-table th,
.orders-table th,
.contacts-table th {
    background-color: #f2f2f2;
    font-weight: 600;
}
.products-admin-table img {
    max-width: 50px;
    max-height: 50px;
    object-fit: cover;
    border-radius: 4px;
}
.products-admin-table .actions button,
.orders-table .save-order-status-btn,
.contacts-table .save-contact-status-btn {
    padding: 5px 10px;
    margin-right: 5px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 0.9em;
}

.products-admin-table    .edit-product-btn    {    background-color:
#ffc107; color: #333; }
.products-admin-table  .edit-product-btn:hover  {  background-color:
#e0a800; }
.products-admin-table   .delete-product-btn   {   background-color:
#dc3545; color: white; }
.products-admin-table .delete-product-btn:hover { background-color:
#c82333; }
```

```
.orders-table .save-order-status-btn,
.contacts-table   .save-contact-status-btn   {   background-color:
#007bff; color: white; }
.orders-table .save-order-status-btn:hover,
.contacts-table .save-contact-status-btn:hover { background-color:
#0056b3; }

.orders-table ul {
    list-style: none;
    padding: 0;
    margin: 0;
    font-size: 0.9em;
}
.orders-table ul li {
    margin-bottom: 3px;
}
#product-edit-form {
    display: none; /* Приховуємо форму за замовчуванням */
    margin-top: 20px;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 8px;
    background-color: #fff;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}
#product-edit-form .form-actions {
    display: flex;
    gap: 10px;
    margin-top: 15px;
}

#product-edit-form .form-actions button {
    margin-top: 0; /* Перевизначити маргін з загальних правил для
кнопок форми */
}
/* Футер */
footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 1em 20px;
    margin-top: 30px;
    font-size: 0.9em;
}
/* Sidebar фільтрів */
main.has-sidebar {
    display: flex;
    gap: 20px;
    align-items: flex-start; /* Вирівнювання елементів по верху */
}
.filters-sidebar {
    flex: 0 0 250px; /* Фіксована ширина */
    background-color: #f9f9f9;
    border: 1px solid #ddd;
    border-radius: 8px;
```

```
    padding: 15px;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.05);
    transition:    all    0.3s    ease;    /*    Анімація    для
схлопування/розгортання */
}
.filters-sidebar.collapsed {
    flex: 0 0 0; /* Сховати sidebar */
    padding: 0;
    overflow: hidden;
    border: none;
    box-shadow: none;
    opacity: 0; /* Для плавного зникнення */
}

main.has-sidebar.sidebar-collapsed {
    margin: 20px auto; /* Центрувати основний контент */
    max-width: 1000px; /* Збільшити ширину для контенту */
}

main.has-sidebar.sidebar-collapsed > section:not(.filters-sidebar)
{
    width: 100%;
    flex-grow: 1;
}


#toggle-sidebar-btn {
    display: block; /* Завжди видима */
    margin-bottom: 20px;
    background-color: #007bff;
    color: white;
    border: none;
    padding: 10px 15px;
    border-radius: 5px;
    cursor: pointer;
}
.filter-group {
    margin-bottom: 20px;
}
.filter-group h4 {
    margin-top: 0;
    margin-bottom: 10px;
    color: #555;
    font-size: 1.1em;
}
.filter-options label {
    display: block;
    margin-bottom: 5px;
    font-weight:   normal;   /*   Відміняємо   жирний   шрифт   для
checkbox/radio label */
}
.filter-actions {
    display: flex;
    flex-direction: column;
    gap: 10px;
```

```
    margin-top: 20px;
}
/* Адаптивність */
@media (max-width: 768px) {
    header {
        flex-direction: column;
        text-align: center;
    }
    header h1, header nav ul {
        order: unset; /* Скидаємо порядок */
    }
    header nav ul {
        margin-top: 10px;
        flex-direction: column;
        gap: 10px;
    }
    main.has-sidebar {
        flex-direction: column; /* На мобільних - колонкою */
    }
    .filters-sidebar {
        flex: 1 1 100%; /* Займає всю ширину */
        width: 100%;
        box-sizing: border-box;
        margin-bottom: 20px;
        order: -1; /* Переміщуємо фільтри нагору */
    }
    main.has-sidebar.sidebar-collapsed   >   section:not(.filters-
sidebar) {
        width: 100%;
        flex-grow: 1;
    }
    .filters-sidebar .search-bar input[type="text"],
    .filters-sidebar .filter-actions button {
        width: 100%;
    }
    .filters-sidebar .search-bar {
        flex-direction: column;
    }
    #product-list-container {
        grid-template-columns:   repeat(auto-fill,   minmax(140px,
1fr));
        gap: 20px;
    }
    .product-card img {
        height: 120px;
    }
    .product-card h3 {
        font-size: 1em;
    }
    .product-card .product-details ul {
        font-size: 0.8em;
    }
    .product-card .price {
        font-size: 1.1em;
    }
```

```
    .product-detail-card {
        flex-direction: column;
        padding: 20px;
    }
    .product-detail-image {
        max-width: 100%;
    }
    .product-info {
        min-width: unset; /* Скидаємо мінімальну ширину */
    }
    .cart-item {
        flex-direction: column;
        align-items: flex-start;
    }
    .cart-item img {
        margin-bottom: 10px;
    }
    .products-admin-table,
    .orders-table,
    .contacts-table {
        font-size: 0.8em; /* Зменшуємо шрифт таблиць */
        display: block; /* Для кращого скролінгу */
        overflow-x: auto; /* Дозволяємо горизонтальний скролінг */
        white-space: nowrap; /* Забороняємо переноси тексту */
    }
}
@media (max-width: 480px) {
    header {
        padding: 1em 10px;
    }
    main {
        padding: 15px;
        margin: 10px auto;
    }
    form {
        padding: 15px;
    }
    #product-list-container {
        grid-template-columns: 1fr; /* Один продукт на рядок */
    }
    .product-card img {
        height: 150px;
    }
}
```

Запит sql для створення таблиць

```
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    is_admin BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

```
);
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL,
    type VARCHAR(100),
    manufacturer VARCHAR(100),
    weight_grams INT NOT NULL,
    image_url VARCHAR(255),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at  TIMESTAMP  DEFAULT  CURRENT_TIMESTAMP  ON  UPDATE
CURRENT_TIMESTAMP
);
CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    customer_name VARCHAR(255) NOT NULL,
    customer_email VARCHAR(255) NOT NULL,
    customer_phone VARCHAR(255) NOT NULL,
    total_amount DECIMAL(10, 2) NOT NULL,
    shipping_address TEXT NOT NULL,
    payment_method VARCHAR(50) NOT NULL,
    delivery_type VARCHAR(50) NOT NULL DEFAULT 'Додому', -- 'Додому'
або 'Відділення Нової Пошти'
    delivery_details TEXT, -- Для номера відділення або додаткових
деталей адреси
    status VARCHAR(50) DEFAULT 'Pending', -- Pending, Processing,
Shipped, Completed, Cancelled
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at  TIMESTAMP  DEFAULT  CURRENT_TIMESTAMP  ON  UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE SET NULL
);
CREATE TABLE order_items (
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    price  DECIMAL(10,  2)  NOT  NULL,  --  Ціна  продукту  на  момент
замовлення
    FOREIGN KEY (order_id) REFERENCES orders(id) ON DELETE CASCADE,
    FOREIGN  KEY  (product_id)  REFERENCES  products(id)  ON  DELETE
RESTRICT
);
CREATE TABLE contacts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    message TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status  VARCHAR(50)  DEFAULT  'New'  --  New,  Replied,  Archived,
Resolved
);
```