

**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: СИСТЕМА УПРАВЛІННЯ ДАНИМИ ДЛЯ МАЛОГО БІЗНЕСУ

Здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи КБ-21-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за галузями)»

(підпис)

Курочки Артура

Олексійовича

(прізвище, ім'я, по батькові)

Науковий керівник

(підпис)

Горбовий Артур

Юліанович, д.т.н., професор

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Керівник
освітньо-професійної
програми

(підпис)

Омельчук Антон

Анатолійович, к.т.н.

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних
технологій і систем протокол від «13» травня 2025 № 18

Завідувач кафедри

(підпис)

Філоненко Михайло

Миколайович, к.ф.-м.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Ірпінь – 2025

Реєстрація № 54

26.05.2025

(дата)



(підпис відповідальної особи)

Результати перевірки

до захисту

(до захисту, на доопрацювання)

26.05.2025

(дата)



(підпис)

Результат захисту

4 (добре)

(оцінка)

05.06.2025

(дата)



(підпис)

Голова комісії:



(підпис)

В.М. Франчук

Члени комісії:



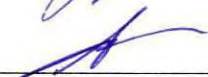
(підпис)

М.М. Філоненко



(підпис)

С.М. Грищенко



(підпис)

А.А. Омельчук

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	21		Горбовий А.Ю.  Франчук В.М.  Філоненко М.М.  Грищенко С.М.  Омельчук А.А.  Суховерхівка Л.П. 
2. Дотримання вимог щодо оформлення	від 0 до 10	8		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	3		
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	37		

Оцінювання кваліфікаційної роботи науковим керівником - 41 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100- бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підпис)
78	C	4 (добре)	05.06.2025	Франчук В.М.  Філоненко М.М.  Грищенко С.М.  Омельчук А.А.  Суховерхівка Л.П. 

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	F X	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Інформаційні управляючі системи і технології (за галузями)»

Затверджую
Завідувач кафедри  Філоненко М.М.
«14» листопада 2024

Завдання на кваліфікаційну роботу
здобувачу вищої освіти
Курочці Артуру Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Система управління даними для малого бізнесу.
Затверджена наказом ректора від «13» листопада 2024 № 1717
2. Строк здачі закінченої роботи «05» травня 2025
3. Мета роботи створення ефективної систему управління даними для малого бізнесу, яка дозволить зберігати, управляти, а також керувати даними.
4. Об'єкт дослідження процеси що пов'язані зі збиранням, зберіганням та використанням інформації для щоденної роботи та прийняття рішень.
Предмет дослідження інструменти, які використовуються для управління базами даних(УБД), побудова БД і способи взаємодії з інтерфейсом програми.
Орієнтовна база дослідження рішення у сфері CRM та ERP-систем, що застосовуються в малому бізнесі, а також технології для створення веб-застосунків. Системи керування БД(MySQL, PostgreSQL), а також фреймворки для розробки веб-інтерфейсів - Vue.js, React.
5. Перелік графічного матеріалу зображення інтерфейсу системи, схеми, які показують як працює система, ключові блоки коду, діаграми зв'язків між частинами системи та базою даних.
6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ СИСТЕМ УПРАВЛІННЯ ДАНИМИ ДЛЯ МАЛОГО БІЗНЕСУ

- 1.1. Аналіз сучасних вимог до систем управління даними в сегменті малого бізнесу
- 1.2. Огляд технологій для розробки веб-орієнтованих систем управління даними (HTML, CSS, PHP)
- 1.3. Особливості проектування баз даних для малого бізнесу з використанням phpMyAdmin

РОЗДІЛ 2. Проектування системи управління даними для малого бізнесу

- 2.1. Визначення функціональних вимог до системи управління даними
- 2.2. Розробка архітектури системи та проектування бази даних
- 2.3. Проектування інтерфейсу користувача та взаємодії компонентів системи

РОЗДІЛ 3. Реалізація та тестування системи управління даними

- 3.1. Розробка frontend-компонентів системи з використанням HTML і CSS
 3.2. Реалізація backend-логіки на PHP та інтеграція з базою даних phpMyAdmin
 3.3. Тестування, оцінка ефективності та рекомендації щодо впровадження системи

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	до 15.11.2024	«14» листопада 2024
2	I розділ (Теоретичні основи систем управління даними для малого бізнесу)	30.01.2025	«30» січня 2025
3	II розділ (Проектування системи управління даними для малого бізнесу)	28.02.2025	«28» лютого 2025
4	III розділ (Реалізація та тестування системи управління даними)	28.04.2025	«26» квітня 2025
5	Висновки	28.04.2025	«27» квітня 2025
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	23.05.2025	«09» травня 2025
7	Отримання відгуку керівника	23.05.2025	«12» травня 2025
8	Отримання зовнішньої рецензії	28.05.2025	«28» травня 2025

8. Дата видачі завдання «14» листопада 2024

9. Керівник кваліфікаційної роботи


(підпис)

Горбовий А.І.

10. Керівник освітньо-професійної програми


(підпис)

Омельчук А.

11. Завдання до виконання отримав здобувач
 вищої освіти


(підпис)

Курочка А.О.

АНОТАЦІЯ

Розробка системи управління даними для малого бізнесу на прикладі кав'ярні.

Мета роботи полягає в розробці веб-орієнтованої системи для автоматизації управління даними кав'ярні, підвищення ефективності бізнес-процесів і якості обслуговування. Завданнями роботи є:

- Аналіз вимог до систем управління даними.
- Проектування трирівневої архітектури MVC.
- Розробка реляційної бази даних у MySQL.
- Створення адаптивного інтерфейсу з HTML, CSS (Flexbox).
- Реалізація backend-логіки на PHP із захистом від SQL-ін'єкцій.
- Тестування системи та оцінка її ефективності.

Предметом дослідження є розробка веб-системи управління даними.

Об'єктом дослідження є процеси управління даними в малому бізнесі.

Базою дослідження виступає діяльність кав'ярні.

За результатами дослідження створено систему, яка автоматизує управління клієнтами, продуктами, замовленнями, забезпечує фільтрацію даних і аналіз.

Одержані результати можуть бути використані для оптимізації роботи кав'ярень та інших малих підприємств.

Кваліфікаційна робота містить 60 сторінок, 13 рисунків, 3 таблиці, список використаних джерел 30 найменувань, 3 додатка.

Ключові слова: система управління, малий бізнес, кав'ярня, HTML, CSS, PHP, MySQL, MVC.

Рік виконання кваліфікаційної роботи: 2024

Рік захисту роботи: 2025

ABSTRACT

Development of a Data Management System for Small Businesses Using a Café as an Example

The goal is to develop a web-based system to automate data management in a café, enhancing business process efficiency and service quality. Tasks include:

- Analyzing data management system requirements.
- Designing a three-tier MVC architecture.
- Developing a relational MySQL database.
- Creating an adaptive interface with HTML, CSS (Flexbox).
- Implementing PHP backend logic with SQL injection protection.
- Testing and evaluating system performance.

The subject is the development of a web-based data management system.

The object is data management processes in small businesses.

The research base is café operations.

The results include a system automating client, product, and order management, with data filtering and analysis.

The findings can optimize café operations and other small businesses.

The thesis comprises 60 pages, 13 figures, 3 tables, 30 sources, 3 appendices.

Keywords: data management, small business, café, HTML, CSS, PHP, MySQL, MVC.

Year of completion: 2025

Year of defense: 2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ.....	8
ВСТУП.....	9
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ СИСТЕМ УПРАВЛІННЯ ДАНИМИ ДЛЯ МАЛОГО БІЗНЕСУ.....	14
1.1. Аналіз сучасних вимог до систем управління даними в сегменті малого бізнесу.....	14
1.2. Огляд технологій для розробки веб-орієнтованих систем управління даними (HTML, CSS, PHP).....	17
1.3. Особливості проектування баз даних для малого бізнесу з використанням phpMyAdmin	23
РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ДАНИМИ ДЛЯ МАЛОГО БІЗНЕСУ	25
2.1. Визначення функціональних вимог до системи управління даними.....	25
2.2. Розробка архітектури системи та проектування бази даних.....	29
2.3. Проектування інтерфейсу користувача та взаємодії компонентів системи.....	33
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ДАНИМИ.....	39
3.1. Розробка frontend-компонентів системи з використанням HTML і CSS.....	39
3.2. Реалізація backend-логіки на PHP та інтеграція з базою даних phpMyAdmin.....	45
3.3. Тестування, оцінка ефективності та рекомендації щодо впровадження системи.....	52
ВИСНОВКИ.....	58
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	60
ДОДАТКИ.....	65

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

Apache — це вільно поширюваний веб-сервер із відкритим вихідним кодом, офіційно відомий як Apache HTTP Server.

API (Application Programming Interface, інтерфейс програмування додатків) — це набір правил, інструментів і визначень, який дозволяє різним програмам чи компонентам обмінюватися даними та взаємодіяти між собою.

MVC (Model-View-Controller) — це архітектурний шаблон (патерн) проєктування, який використовується для організації коду в програмуванні, зокрема при розробці веб-додатків. Він розділяє логіку програми на три взаємодіючі компоненти, що полегшує підтримку, масштабування та тестування.

XML (Extensible Markup Language, розширювана мова розмітки) — це універсальний формат для структурованого представлення даних, який використовується для обміну інформацією між різними системами та програмами. Це текстовий формат, який дозволяє створювати власні теги для опису даних, що робить його гнучким і придатним для різних застосувань.

UML (Unified Modeling Language, уніфікована мова моделювання) — це стандартизована мова для створення діаграм, які використовуються для моделювання, документування та аналізу програмних систем. Вона допомагає розробникам візуалізувати структуру, поведінку та взаємодію компонентів системи ще на етапі проєктування,

CRUD-операції — це базовий набір функцій для роботи з даними в інформаційних системах, зокрема в базах даних. Назва є акронімом від англійських слів: Create, Read, Update, Delete, що відповідають основним діям із даними.

ВСТУП

Актуальність теми: у сучасному бізнес-середовищі ефективне управління даними стало визначальним фактором конкурентоспроможності підприємств будь-якого масштабу, включаючи суб'єкти малого бізнесу. Особливу актуальність ця проблема має для закладів громадського харчування, зокрема кав'ярень, які щоденно генерують значні обсяги інформації про клієнтів, продукти, замовлення та фінансові операції. Традиційні методи збору та обробки даних, такі як паперові журнали обліку чи таблиці Excel, вже не відповідають сучасним вимогам бізнесу щодо швидкості, точності та комплексності аналізу інформації. Водночас, повномасштабні комерційні системи управління ресурсами підприємства (ERP) часто є надто складними та дорогими для впровадження в малому бізнесі. Це створює потребу в розробці спеціалізованих, економічно доступних та функціонально оптимізованих систем управління даними, що враховують специфіку роботи малих підприємств сфери громадського харчування. Використання веб-технологій для реалізації таких систем забезпечує їх доступність з різних пристроїв без необхідності встановлення спеціального програмного забезпечення, що особливо цінні для малого бізнесу з обмеженими ресурсами на IT-інфраструктуру.

Мета і завдання дослідження: метою дипломної роботи є дослідження теоретичних та практичних аспектів розробки системи управління даними для малого бізнесу та створення функціональної веб-орієнтованої системи для кав'ярні з використанням технологій HTML, CSS та PHP.

Для досягнення поставленої мети визначено наступні завдання:

1. Проаналізувати сучасні вимоги до систем управління даними в сегменті малого бізнесу.
2. Дослідити технології для розробки веб-орієнтованих систем управління даними.
3. Визначити особливості проектування баз даних для малого бізнесу з використанням phpMyAdmin.

4. Розробити функціональні вимоги до системи управління даними для кав'ярні.
5. Спроекувати архітектуру системи та структуру бази даних.
6. Реалізувати frontend-компоненти системи з використанням HTML і CSS.
7. Розробити backend-логіку системи на PHP та інтегрувати її з базою даних phpMyAdmin.
8. Провести тестування системи, оцінити її ефективність та розробити рекомендації щодо впровадження.

Об'єкт дослідження: процес розробки та впровадження системи управління даними для малого бізнесу в сфері громадського харчування.

Предмет дослідження : методи, технології та інструменти розробки веб-орієнтованої системи управління даними для кав'ярні з використанням HTML, CSS та PHP.

Практичне значення роботи : полягає в розробці функціональної системи управління даними, яка може бути безпосередньо впроваджена в діяльність кав'ярень та інших закладів громадського харчування малого формату. Реалізована система дозволяє автоматизувати провідні бізнес-процеси, такі як управління клієнтською базою, асортиментом продуктів та замовленнями, що сприяє підвищенню ефективності роботи закладу, покращенню якості обслуговування клієнтів та оптимізації ресурсів підприємства. Розроблені в рамках дослідження методи та підходи можуть бути використані як основа для створення аналогічних систем для інших сфер малого бізнесу.

Методи дослідження: у роботі використано комплекс загальнонаукових та спеціальних методів дослідження: аналіз та синтез при вивченні теоретичних аспектів систем управління даними, порівняльний аналіз при оцінці різних технологій розробки, системний підхід при проектуванні архітектури системи, моделювання при створенні структури бази даних, експериментальне дослідження при тестуванні системи, а також методи об'єктно-орієнтованого програмування при розробці програмного коду системи.

Структура роботи: дипломна робота складається зі вступу, трьох розділів, висновків та списку використаних джерел. У першому розділі розглядаються теоретичні основи систем управління даними для малого бізнесу, включаючи аналіз сучасних вимог, огляд технологій розробки та особливості проектування баз даних. Другий розділ присвячено проектуванню системи управління даними для кав'ярні, включаючи визначення функціональних вимог, розробку архітектури системи та проектування інтерфейсу користувача. У третьому розділі описано процес реалізації системи, включаючи розробку frontend-компонентів, реалізацію backend-логіки, а також результати тестування та рекомендації щодо впровадження системи.

Результати дослідження апробовано: II Міжнародній науково-практичній конференції студентів та молодих учених "Інформаційні технології в економіці та управлінні" (м. Ірпінь, Україна, 15–16 квітня 2025 р.). Представлено тези доповіді на тему "Розробка веб-орієнтованої системи управління даними для кав'ярні на основі HTML, CSS, PHP та MySQL", де висвітлено функціонал системи, трирівневу архітектуру MVC, адаптивний інтерфейс із використанням Flexbox та захист від SQL-ін'єкцій. Тези опубліковано у збірнику матеріалів конференції [14]. Результати отримали схвалення учасників конференції, що підтверджує їхню наукову та практичну цінність. У кваліфікаційній роботі використано 3 таблиці, 13 рисунків та 3 додатка.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ СИСТЕМ УПРАВЛІННЯ ДАНИМИ ДЛЯ МАЛОГО БІЗНЕСУ

1.1. Аналіз сучасних вимог до систем управління даними в сегменті малого бізнесу

У сучасному діловому середовищі системи управління даними становлять фундаментальну основу для ефективного функціонування малого бізнесу. Під системою управління даними розуміється комплекс програмних засобів, що забезпечують збирання, зберігання, обробку, аналіз та представлення інформації, необхідної для прийняття бізнес-рішень. Малий бізнес, який традиційно обмежений у ресурсах порівняно з великими корпораціями, потребує особливо ретельного підходу до вибору та впровадження таких систем. Дослідження показують, що оптимально налаштована система управління даними може підвищити продуктивність малого підприємства на 20-30%, що робить її впровадження не просто бажаним, а необхідним кроком для забезпечення конкурентоспроможності [1].

Однією з першочергових вимог до сучасних систем управління даними в сегменті малого бізнесу є їх економічна доступність при збереженні достатньої функціональності. Фінансові обмеження змушують малі підприємства шукати рішення, які надають суттєву користь при мінімальних витратах на впровадження та підтримку. Саме тому веб-орієнтовані системи, розроблені з використанням відкритих технологій, таких як HTML, CSS та PHP, з підтримкою безкоштовних систем управління базами даних, зокрема через інтерфейс phpMyAdmin, набувають особливої популярності. Згідно з даними аналітичних агентств, понад 65% малих підприємств віддають перевагу саме таким рішенням, оскільки вони дозволяють оптимізувати витрати і при цьому отримати гнучкий інструмент для роботи з бізнес-інформацією.

Масштабованість і гнучкість системи становлять наступний суттєвий аспект вимог до систем управління даними для малого бізнесу. Підприємства цього сегменту характеризуються динамічним розвитком, що передбачає поступове збільшення обсягів даних та розширення функціональних потреб

системи. Тому архітектура системи повинна забезпечувати можливість легкого нарощування потужностей без необхідності кардинальної перебудови. Модульний підхід до проектування, коли система складається з окремих функціональних блоків, які можна додавати або модифікувати незалежно, дозволяє адаптувати рішення під зростаючі потреби бізнесу. Цей підхід надзвичайно доречний для малих підприємств, оскільки дозволяє починати з базового набору функцій і поступово розширювати систему в міру необхідності та наявності ресурсів [2].

Безпека даних є одним із найсуттєвіших аспектів при проектуванні систем управління даними незалежно від масштабу бізнесу. Для малих підприємств ця вимога має особливе значення, оскільки втрата або компрометація даних може призвести до катастрофічних наслідків, аж до повного краху бізнесу. Сучасні вимоги включають багаторівневу систему захисту, що охоплює шифрування даних, механізми аутентифікації та авторизації користувачів, регулярне резервне копіювання, а також протоколювання всіх дій у системі. Статистика свідчить, що понад 60% малих підприємств, які зазнали серйозного порушення безпеки даних і не мали налагодженої системи резервного копіювання, припиняють своє існування протягом шести місяців після інциденту. Така сумна статистика підкреслює необхідність впровадження надійних механізмів захисту навіть у найпростіших системах для малого бізнесу. У зв'язку з обмеженими ресурсами та високими ризиками, що супроводжують діяльність малих підприємств, особливо важливим є впровадження систем управління даними, які відповідають специфічним вимогам цієї категорії бізнесу. У таблиці 1.1 узагальнено ключові вимоги до таких систем, що враховують як функціональні, так і організаційні аспекти їх впровадження та використання. Інтуїтивно зрозумілий користувацький інтерфейс є ще однією першорядною вимогою, яка виходить на перший план саме в контексті малого бізнесу. На відміну від великих корпорацій, які можуть дозволити собі утримання штату ІТ-спеціалістів, малі підприємства зазвичай обмежені в можливостях залучення технічних експертів.

Таблиця 1.1

Ключові вимоги до систем управління даними для малого бізнесу

Вимога до системи управління даними	Опис	Значення для малого бізнесу
Економічна ефективність	Оптимальне співвідношення функціональності та вартості впровадження	Дозволяє мінімізувати початкові інвестиції та сукупну вартість володіння
Масштабованість	Можливість нарощування потужностей та функціональності	Забезпечує адаптацію системи до зростаючих потреб бізнесу
Безпека даних	Комплексний захист від несанкціонованого доступу та втрати даних	Гарантує збереження бізнес-інформації та захист від можливих загроз
Інтуїтивний інтерфейс	Зрозуміла взаємодія, що не потребує спеціальної підготовки	Мінімізує витрати на навчання персоналу та прискорює впровадження
Автоматизація процесів	Зменшення ручних операцій з даними	Підвищує продуктивність та знижує ймовірність помилок

Джерело [авторська]

Отже, система управління даними повинна бути розроблена таким чином, щоб пересічний користувач без спеціальної підготовки міг ефективно взаємодіяти з нею. Це досягається завдяки логічній організації елементів інтерфейсу, використанню зрозумілої термінології, наявності контекстних підказок та детальної документації. Дослідження користувацького досвіду показують, що добре спроектований інтерфейс може зменшити час на виконання типових операцій на 30-40%, що безпосередньо впливає на загальну ефективність роботи підприємства [3].

Інтеграційні можливості систем управління даними набувають все більшого значення в умовах цифрової трансформації бізнесу. Сучасний малий бізнес рідко обмежується використанням єдиної інформаційної системи; натомість, підприємства зазвичай застосовують набір спеціалізованих рішень для різних аспектів діяльності – від бухгалтерського обліку до взаємодії з клієнтами. Тому життєво необхідною стає можливість безперешкодного обміну даними між різними системами, що забезпечується наявністю відкритих API (Application Programming Interface) та підтримкою стандартизованих форматів даних. Інтеграція з популярними сервісами електронної комерції, платіжними системами, службами доставки та іншими зовнішніми сервісами дозволяє

створити єдиний інформаційний простір і уникнути проблеми "інформаційних островів", коли дані залишаються ізольованими в окремих системах.

Аналітичні можливості становлять заключний, але не менш визначальний аспект сучасних вимог до систем управління даними для малого бізнесу. В умовах зростаючої конкуренції здатність перетворювати накопичені дані на корисні бізнес-інсайти стає головним фактором успіху. Система повинна надавати інструменти для аналізу історичних даних, виявлення трендів, прогнозування майбутніх показників та підтримки прийняття рішень. Візуалізація даних у вигляді інтерактивних графіків, діаграм та інформаційних панелей (дашбордів) допомагає керівникам малого бізнесу швидко оцінювати поточний стан справ та приймати обґрунтовані рішення. Дослідження показують, що підприємства, які активно використовують аналітичні інструменти, демонструють на 15-25% вищі показники рентабельності порівняно з конкурентами, які спираються переважно на інтуїтивні рішення. Таким чином, наявність аналітичного модуля в системі управління даними перетворюється з додаткової опції на обов'язкову вимогу для забезпечення конкурентоспроможності малого бізнесу в сучасних умовах [4].

1.2. Огляд технологій для розробки веб-орієнтованих систем управління даними (HTML, CSS, PHP)

Розробка веб-орієнтованих систем управління даними вимагає комплексного застосування різних технологій, кожна з яких виконує специфічну роль у створенні функціонального та естетично привабливого продукту. В основі таких систем зазвичай лежить тріада технологій: HTML для структурування контенту, CSS для оформлення візуального представлення та PHP для забезпечення бізнес-логіки та взаємодії з базами даних. Ця комбінація технологій утворює фундамент, на якому будуються сучасні веб-системи, включно з рішеннями для малого бізнесу. Відповідно до визначення Консорціуму Всесвітньої Павутини (W3C), HTML (HyperText Markup Language) – це

стандартизована мова розмітки документів для перегляду веб-сторінок у браузері. Саме HTML дозволяє структурувати контент, визначаючи, які елементи будуть відображатися на сторінці та як вони будуть організовані [5].

HTML є основою будь-якої веб-системи, забезпечуючи семантичну структуру веб-сторінок та визначаючи їхнє інформаційне наповнення. Сучасний стандарт HTML5, затверджений у 2014 році, суттєво розширив функціональні можливості мови розмітки, включивши підтримку мультимедійного контенту (аудіо, відео), новітніх форм введення даних та семантичних елементів. Останні мають особливе значення для систем управління даними, оскільки дозволяють більш чітко структурувати інформацію, що полегшує її обробку та пошук. Наприклад, використання елементів `<header>`, `<footer>`, `<nav>`, `<section>`, `<article>` замість універсальних `<div>` надає сторінці семантичної виразності, що позитивно впливає не лише на доступність контенту для кінцевих користувачів, а й на індексацію системи пошуковими роботами. Це набуває значущості для малого бізнесу, оскільки підвищує видимість його веб-ресурсів у пошукових системах без необхідності додаткових витрат на SEO-оптимізацію.

Каскадні таблиці стилів (CSS, Cascading Style Sheets) – технологія, що визначає зовнішній вигляд HTML-елементів та їхню поведінку при відображенні у браузері. Відповідно до фундаментального принципу розділення контенту та представлення, CSS відповідає за візуальне оформлення веб-сторінки, тоді як HTML забезпечує структуру контенту. Сучасні стандарти CSS3 надають розробникам потужний інструментарій для створення привабливого користувацького інтерфейсу, включаючи анімації, переходи, градієнти, тіні, заокруглені кути та інші візуальні ефекти без необхідності використання зображень або сторонніх плагінів. Особливо значущою інновацією CSS3 стало впровадження медіа-запитів (media queries), які дозволяють адаптувати відображення сторінки до різних пристроїв та розмірів екранів. Це надзвичайно суттєво для малого бізнесу, оскільки забезпечує доступність системи управління даними на різних пристроях – від настільних комп'ютерів до смартфонів – без

необхідності розробки окремих версій інтерфейсу для кожного типу пристрою, що дозволяє оптимізувати бюджет на розробку та підтримку системи [6].

У сфері розробки веб-орієнтованих систем управління даними для малого бізнесу особливе місце посідає PHP (рекурсивний акронім від "PHP: Hypertext Preprocessor") – скриптова мова програмування з відкритим вихідним кодом, спеціально розроблена для створення динамічних веб-сторінок. Згідно з визначенням, наведеним в офіційній документації PHP, це мова, яка виконується на стороні сервера і вбудовується безпосередньо в HTML-код, що дозволяє швидко генерувати динамічний контент. Основною перевагою PHP є його нативна інтеграція з більшістю сучасних систем управління базами даних, включаючи MySQL, PostgreSQL, SQLite та інші, що робить цю мову надзвичайно ефективним інструментом для створення систем, що оперують значними обсягами даних. Додатковою перевагою PHP є наявність величезної кількості готових бібліотек та фреймворків, таких як Laravel, Symfony, CodeIgniter та інші, які значно прискорюють процес розробки завдяки реалізації шаблону MVC (Model-View-Controller), що забезпечує чітке розділення бізнес-логіки, даних та представлення, полегшуючи подальшу підтримку та масштабування системи [7].

Для забезпечення ефективної взаємодії між описаними технологіями, в сучасній веб-розробці широко застосовуються JavaScript та його бібліотеки, зокрема jQuery, React, Vue.js та інші. JavaScript – мова програмування, що виконується на стороні клієнта (в браузері користувача) і дозволяє створювати інтерактивні елементи інтерфейсу, обробляти дії користувача без перезавантаження сторінки, валідувати дані форм перед відправкою на сервер та здійснювати асинхронний обмін даними з сервером за допомогою технології AJAX (Asynchronous JavaScript and XML). В контексті систем управління даними для малого бізнесу, JavaScript забезпечує оперативну взаємодію з користувачем, миттєве оновлення інтерфейсу при зміні даних та можливість офлайн-роботи з частиною функціоналу системи, що суттєво підвищує зручність користування та продуктивність роботи персоналу. Особливо варто відзначити бібліотеки для створення інтерактивних таблиць та візуалізації даних, такі як DataTables,

Chart.js, D3.js, які дозволяють перетворити сухі цифри на наочні графіки та діаграми, що спрощує аналіз інформації та прийняття управлінських рішень.

Невід'ємним компонентом розробки веб-орієнтованих систем управління даними є серверні технології та середовища розгортання. Для PHP-додатків найбільш поширеним є використання веб-серверів Apache або Nginx у поєднанні з операційними системами Linux, що утворює так званий стек LAMP (Linux, Apache, MySQL, PHP) або LEMP (Linux, Nginx, MySQL, PHP). Ці технологічні стеки пропонують оптимальне поєднання продуктивності, безпеки та масштабованості при відносно низьких вимогах до апаратних ресурсів, що робить їх ідеальним вибором для малого бізнесу. Альтернативою традиційному хостингу є використання хмарних платформ, таких як Amazon Web Services (AWS), Microsoft Azure або Google Cloud Platform, які пропонують послуги PaaS (Platform as a Service) та SaaS (Software as a Service), дозволяючи розгорнути веб-системи без необхідності конфігурування та підтримки серверної інфраструктури. Цей підхід особливо привабливий для малого бізнесу, оскільки дозволяє гнучко масштабувати обчислювальні ресурси відповідно до потреб і сплачувати лише за фактично використані ресурси, уникаючи суттєвих початкових капіталовкладень у серверне обладнання [8].

Для забезпечення безпеки веб-орієнтованих систем управління даними застосовується комплекс технологій та методологій, включаючи шифрування даних за допомогою протоколу HTTPS (HTTP Secure), використання підписаних SSL/TLS сертифікатів, захист від SQL-ін'єкцій через підготовлені запити (prepared statements) або ORM (Object-Relational Mapping) в PHP, запобігання XSS-атакам (Cross-Site Scripting) шляхом екранування вхідних даних, захист від CSRF-атак (Cross-Site Request Forgery) за допомогою токенів, а також реалізацію надійної системи аутентифікації та авторизації користувачів з використанням хешування паролів та багатофакторної аутентифікації. Особливу увагу слід приділяти регулярному оновленню використовуваних фреймворків та бібліотек, оскільки через їх популярність вони часто стають мішенню для злоумисників, а розробники оперативно випускають патчі(оновлення) для виявлених

вразливостей. Для малого бізнесу впровадження належних практик безпеки має першочергове значення, оскільки інциденти безпеки можуть призвести до значних фінансових втрат та втрати довіри клієнтів, що може бути фатальним для бізнесу з обмеженими ресурсами.

Вибір конкретних технологій та інструментів для розробки веб-орієнтованої системи управління даними повинен ґрунтуватися на специфічних потребах конкретного малого бізнесу, наявних ресурсах та перспективах масштабування. В загальному випадку, оптимальним рішенням для малого бізнесу є використання PHP у поєднанні з фреймворком середнього рівня абстракції (наприклад, Laravel або Symfony), який забезпечує достатню гнучкість та функціональність при помірній складності розробки. Для користувацького інтерфейсу доцільно застосовувати адаптивні CSS-фреймворки, такі як Bootstrap або Foundation, які дозволяють швидко створювати естетично привабливі та функціональні інтерфейси, що коректно відображаються на різних пристроях. Таке технологічне рішення забезпечує оптимальний баланс між функціональністю, зручністю користування, часом розробки та вартістю впровадження, що відповідає основним вимогам малого бізнесу до систем управління даними. Візуалізація даних може бути реалізована за допомогою JavaScript-бібліотек, таких як Chart.js або Highcharts, які пропонують широкий спектр типів графіків та діаграм при відносно простому API, що дозволяє швидко впроваджувати аналітичні функції в систему. Для більш наочного представлення взаємозв'язків між описаними технологіями та компонентами системи, розроблено UML діаграму класів, що демонструє архітектуру веб-орієнтованої системи управління даними (рис. 1.1.) [6].

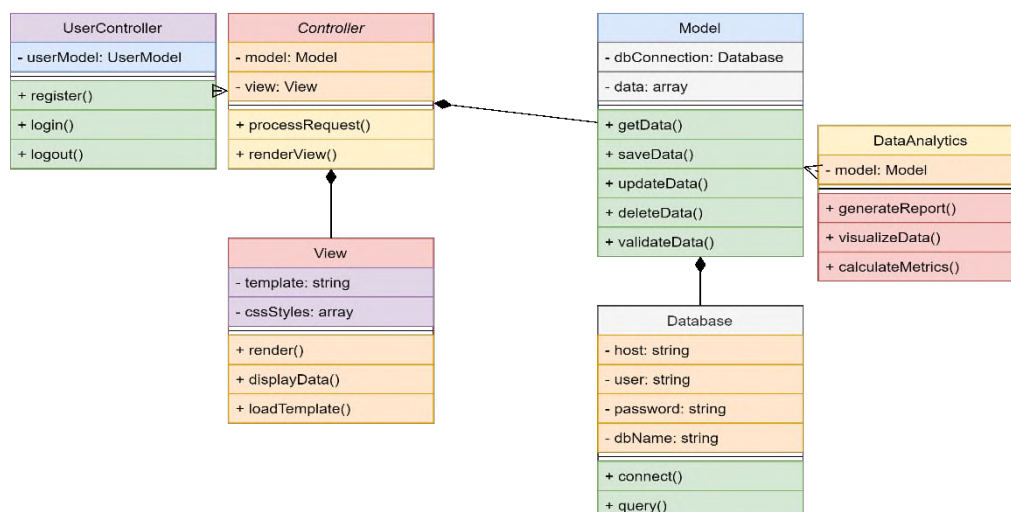


Рис. 1.1 UML діаграма взаємодії компонентів веб-орієнтованої системи управління даними

Представлена UML діаграма демонструє архітектурне рішення для системи управління даними малого бізнесу на основі розглянутих технологій. Центральним елементом архітектури виступає шаблон проектування MVC (Model-View-Controller), де контролери обробляють вхідні запити користувачів, моделі забезпечують бізнес-логіку та взаємодію з базою даних, а представлення відповідають за відображення інформації кінцевим користувачам. Така структура дозволяє досягти чіткого розділення відповідальності між компонентами системи, що суттєво спрощує подальший розвиток та підтримку програмного продукту. Завдяки використанню HTML для структурування контенту, CSS для стилізації інтерфейсу та PHP для реалізації серверної логіки, система здатна забезпечити оптимальний баланс між функціональністю, продуктивністю та зручністю використання. Додаткові компоненти, такі як модуль аналітики (DataAnalytics), дозволяють розширити базову функціональність системи для задоволення специфічних потреб малого бізнесу, зокрема в частині візуалізації даних та підтримки прийняття управлінських рішень. Таким чином, раціональне поєднання розглянутих технологій створює надійний фундамент для розробки ефективних веб-орієнтованих систем

управління даними, що повною мірою відповідають вимогам сучасного малого бізнесу.

1.3. Особливості проектування баз даних для малого бізнесу з використанням phpMyAdmin

Проектування баз даних для малого бізнесу характеризується низкою специфічних особливостей, зумовлених як масштабом діяльності, так і обмеженістю ресурсів підприємств даного сегменту. Під базою даних у контексті інформаційних систем розуміється структурована сукупність даних, організованих відповідно до концептуальної схеми, що описує характеристики цих даних і взаємозв'язки між ними. Для малого бізнесу оптимальним рішенням часто виступають реляційні бази даних, які забезпечують належний баланс між функціональними можливостями та простотою впровадження. В процесі проектування таких баз даних суттєвого значення набуває вибір інструментарію, що дозволить ефективно створювати та підтримувати інформаційну структуру з мінімальними затратами. Одним із найпоширеніших інструментів для роботи з базами даних MySQL є phpMyAdmin – веб-інтерфейс для адміністрування СУБД MySQL, написаний на мові PHP. Згідно з офіційним визначенням, phpMyAdmin представляє собою безкоштовний програмний інструмент, що надає повноцінний веб-інтерфейс для керування базами даних MySQL та MariaDB, дозволяючи виконувати найпоширеніші операції (створення, модифікація, видалення баз даних, таблиць, полів та рядків, виконання SQL-запитів) через зручний графічний інтерфейс [9].

Визначальною перевагою використання phpMyAdmin при проектуванні баз даних для малого бізнесу є суттєве спрощення процесу адміністрування за рахунок наявності інтуїтивно зрозумілого візуального інтерфейсу. На відміну від великих підприємств, які можуть дозволити собі утримувати штат кваліфікованих адміністраторів баз даних, у сегменті малого бізнесу роботу з інформаційними системами зазвичай виконують співробітники з обмеженим

досвідом у сфері інформаційних технологій. В таких умовах особливо доречним стає використання phpMyAdmin, який не потребує глибоких знань синтаксису SQL для виконання базових операцій з базою даних. Візуальний конструктор запитів, редактор таблиць та інші графічні інструменти значно полегшують процес створення та модифікації структури бази даних, що дозволяє малим підприємствам заощаджувати на витратах для залучення високооплачуваних фахівців. Водночас, phpMyAdmin не обмежує можливості досвідчених користувачів, надаючи їм повний доступ до виконання довільних SQL-запитів, що забезпечує гнучкість при реалізації специфічних потреб бізнесу.

При проектуванні баз даних для малого бізнесу з використанням phpMyAdmin першочергового значення набуває раціональне структурування інформації з урахуванням специфіки діяльності конкретного підприємства. На етапі концептуального проектування визначаються базові сутності, їхні атрибути та взаємозв'язки між ними. Для малого бізнесу типовими сутностями зазвичай виступають: клієнти, товари/послуги, замовлення, постачальники, працівники, фінансові транзакції тощо. Особливістю баз даних для малого бізнесу є необхідність забезпечення балансу між нормалізацією та продуктивністю. Однак надмірна нормалізація може призвести до ускладнення запитів та зниження продуктивності системи. Для малого бізнесу оптимальним зазвичай є досягнення третьої нормальної форми (3НФ), яка забезпечує достатній рівень цілісності даних при збереженні прийнятної продуктивності. Інтерфейс phpMyAdmin значно спрощує процес проектування таблиць відповідно до принципів нормалізації, надаючи зручні інструменти для визначення типів даних, первинних та зовнішніх ключів, унікальних індексів та інших обмежень цілісності [10].

Безпека даних становить фундаментальний аспект при проектуванні баз даних, особливо для малого бізнесу, де втрата або компрометація інформації може мати катастрофічні наслідки. PhpMyAdmin надає розгорнуті можливості для забезпечення належного рівня захисту інформації через зручний веб-інтерфейс. Серед них варто відзначити гнучку систему управління

користувачами та їхніми привілеями, що дозволяє диференціювати доступ відповідно до функціональних обов'язків співробітників. Вбудовані механізми автентифікації, включаючи можливість використання двофакторної автентифікації, суттєво підвищують захищеність системи від несанкціонованого доступу.

Ефективне управління даними передбачає регулярне резервне копіювання та можливість швидкого відновлення інформації у випадку збоїв або інших інцидентів. У контексті малого бізнесу, де часто відсутні виділені системи резервного копіювання, інструментарій phpMyAdmin набуває особливої цінності. Вбудовані функції експорту даних дозволяють створювати резервні копії як окремих таблиць, так і всієї бази даних у різних форматах (SQL, CSV, XML, JSON тощо). Особливо корисною для малого бізнесу є можливість автоматизації процесу резервного копіювання шляхом планування регулярного експорту даних. Для цього можна використовувати стандартні можливості операційної системи (наприклад, планувальник завдань Windows або cron у Linux) у поєднанні з командним інтерфейсом phpMyAdmin. Функціональність імпорту даних забезпечує швидке відновлення інформації з резервних копій, а також спрощує процес міграції з інших систем управління базами даних. Наприклад, малий бізнес, який раніше використовував електронні таблиці для ведення обліку, може легко імпортувати ці дані у структуровану базу даних через phpMyAdmin, що значно підвищує ефективність обробки інформації.

Продуктивність бази даних є визначальним фактором, що впливає на швидкодію всієї інформаційної системи малого бізнесу. PhpMyAdmin пропонує низку інструментів для оптимізації продуктивності, доступних через зручний веб-інтерфейс. Вбудовані засоби аналізу та оптимізації запитів дозволяють виявляти неефективні SQL-вирази та покращувати їх структуру для прискорення виконання. Система моніторингу продуктивності надає детальну інформацію про час виконання запитів, використання пам'яті та інші параметри, що дозволяє ідентифікувати вузькі місця та оперативно реагувати на них. Особливу увагу при проектуванні баз даних для малого бізнесу слід приділяти створенню

оптимальних індексів, які суттєво прискорюють пошук та сортування даних. PhpMyAdmin надає зручний інтерфейс для управління індексами, включаючи створення складених індексів та показ статистики їх використання. Для таблиць з великими обсягами даних (наприклад, історія транзакцій, журнали активності) доцільним є використання механізму партиціонування, який також підтримується через інтерфейс phpMyAdmin. Партиціонування дозволяє розбити велику таблицю на менші логічні частини, що значно покращує продуктивність запитів, спрощує адміністрування та забезпечує більш ефективне використання дискового простору [12].

Проектування бази даних для малого бізнесу неодмінно повинно враховувати перспективи масштабування системи відповідно до зростання підприємства. В цьому контексті phpMyAdmin пропонує ряд функціональних можливостей, що забезпечують гнучкість та адаптивність структури даних. Зокрема, підтримка зберігаємих процедур, тригерів та функцій дозволяє інкапсулювати бізнес-логіку на рівні бази даних, що спрощує подальше розширення функціональності системи без необхідності внесення змін у прикладний код. Механізм подій (events) забезпечує можливість планування та автоматичного виконання регулярних операцій з базою даних, таких як архівування старих даних, формування періодичних звітів, оновлення статистичної інформації тощо. Підтримка представлень (views) дозволяє створювати віртуальні таблиці, що відображають дані у зручному для користувачів форматі без дублювання інформації, що особливо корисно при розробці аналітичних модулів системи. При проектуванні бази даних для малого бізнесу з використанням phpMyAdmin доцільно також враховувати можливість інтеграції з іншими системами та сервісами, що досягається завдяки підтримці стандартних форматів даних та протоколів обміну інформацією. Така інтеграційна спроможність стає дедалі більш значущою в умовах цифрової трансформації бізнесу та зростаючої ролі електронної комерції, особливо для малих підприємств, які прагнуть розширити свою присутність на ринку [10].

РОЗДІЛ 2. ПРОЕКТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ДАНИМИ ДЛЯ МАЛОГО БІЗНЕСУ

2.1. Визначення функціональних вимог до системи управління даними

Функціональні вимоги до системи управління даними для малого бізнесу являють собою формалізований опис очікуваної поведінки системи, що визначає конкретні функції, які система повинна виконувати для задоволення потреб користувачів. Згідно зі стандартом IEEE 830, функціональні вимоги визначаються як "твердження, що описують, що саме система повинна робити, при отриманні яких вхідних даних та при яких умовах". У контексті розроблюваної системи управління даними для кав'ярні, функціональні вимоги базуються на аналізі бізнес-процесів підприємства та специфіки галузі. Проведений аналіз вихідного коду прототипу системи (files: index.php, clients.php, products.php, orders.php та інші) дозволяє ідентифікувати основні сутності, з якими працює система, а саме: клієнти, продукти та замовлення. Всі ці сутності підлягають операціям створення, читання, оновлення та видалення (CRUD-операції), що є фундаментальними функціональними вимогами для будь-якої системи управління даними. Однак, специфіка роботи кав'ярні вимагає додаткових функціональних можливостей, таких як фільтрація та пошук замовлень за різними критеріями, візуалізація каталогу продуктів, автоматичний розрахунок вартості замовлень, а також формування звітності за різними параметрами діяльності закладу [13].

Центральним елементом функціональних вимог до системи є управління клієнтською базою, що передбачає можливість реєстрації нових клієнтів, перегляду інформації про наявних клієнтів та її редагування, а також видалення клієнтів у разі потреби. Аналіз файлів add_client.php та clients.php (Додаток А) показує, що система вже реалізує базові функції для роботи з клієнтами, такі як додавання нового клієнта з фіксацією його ПІБ, електронної пошти та телефону, а також перегляд списку клієнтів з можливістю видалення. Водночас, для забезпечення повноцінної роботи кав'ярні доцільно розширити ці функціональні

вимоги, додавши можливість категоризації клієнтів (наприклад, за частотою відвідувань або середнім чеком), впровадження програми лояльності з нарахуванням бонусів, ведення історії замовлень конкретного клієнта з можливістю швидкого повторного замовлення його улюблених напоїв або страв. Суттєвим доповненням також може стати функція автоматичного збору та аналізу відгуків клієнтів для покращення сервісу. Ці розширені функціональні вимоги до роботи з клієнтською базою спрямовані на підвищення якості обслуговування та формування стійкої бази постійних клієнтів, що є вирішальним фактором для успішної діяльності малого бізнесу в сфері громадського харчування.

Управління продуктовою лінійкою є другим фундаментальним аспектом функціональних вимог до системи, що охоплює процеси додавання нових продуктів до асортименту, оновлення інформації про наявні продукти, контроль наявності та видалення продуктів, які більше не пропонуються. Аналіз файлів, зокрема `products.php` (Додаток А), демонструє, що в прототипі системи передбачена можливість перегляду списку продуктів з їхніми основними атрибутами (назва, категорія, ціна, зображення), а також фільтрація продуктів за категорією та ціновим діапазоном. Зокрема, функціональні вимоги повинні включати: можливість управління інгредієнтами та рецептурою продуктів, відстеження рівня запасів інгредієнтів з автоматичним сповіщенням про необхідність поповнення, розрахунок собівартості продуктів на основі вартості інгредієнтів, управління сезонними або тимчасовими пропозиціями, можливість створення комбінованих пропозицій (наприклад, "кава + десерт" за спеціальною ціною). Додатково, система повинна забезпечувати функціонал для налаштування відображення продуктів на веб-сторінці або в мобільному додатку, з можливістю виділення спеціальних пропозицій або новинок. Для наочності представлено таблицю 2.1.

Особливо значущою є функція відстеження популярності різних продуктів на основі аналізу замовлень, що допомагає оптимізувати асортимент та сфокусувати маркетингові зусилля на найбільш потрібних позиціях [14].

Таблиця 2.1.

Основні функціональні вимоги до системи управління даними для кав'ярні

Категорія вимог	Функціональні вимоги	Опис
Управління клієнтами	Реєстрація клієнтів	Додавання нових клієнтів з основними даними (ПІБ, email, телефон)
	Перегляд списку клієнтів	Відображення всіх зареєстрованих клієнтів з можливістю пошуку та фільтрації
	Видалення клієнтів	Можливість видалення клієнтів із системи
	Історія замовлень клієнта	Доступ до повної історії замовлень конкретного клієнта
Управління продуктами	Додавання продуктів	Створення нових позицій у меню з описом, ціною та зображенням
	Категоризація продуктів	Розподіл продуктів за категоріями (напої, десерти тощо)
	Фільтрація продуктів	Можливість пошуку продуктів за категорією, ціною та іншими параметрами
	Управління цінами	Можливість оперативного оновлення цін на продукти
Управління замовленнями	Створення замовлення	Формування нового замовлення з вибором клієнта та продуктів
	Пошук та фільтрація замовлень	Можливість фільтрації замовлень за клієнтом, продуктом, датою
	Видалення замовлень	Можливість видалення помилково створених замовлень
	Статистика замовлень	Генерація звітів за різними параметрами замовлень

Джерело [авторська]

Управління замовленнями є третім невід'ємним компонентом функціональних вимог до системи управління даними для кав'ярні. Аналіз файлів `add_order.php` та `orders.php` свідчить про наявність у прототипі базового функціоналу для створення нових замовлень з вибором клієнта, продукту, кількості та дати, а також перегляду списку замовлень з можливістю фільтрації за клієнтом, продуктом та часовим періодом. Для повноцінного забезпечення бізнес-процесів кав'ярні ці функціональні можливості потребують розширення. Зокрема, система повинна забезпечувати: можливість створення комплексних замовлень, що містять декілька різних продуктів; автоматичний розрахунок загальної вартості замовлення з урахуванням можливих знижок або спеціальних

пропозицій; підтримку різних статусів замовлення (наприклад, "прийнято", "в процесі приготування", "готово до видачі", "завершено"); формування чеків з можливістю їх друку або надсилання клієнту електронною поштою; інтеграцію з платіжними системами для обробки оплати; можливість формування замовлень на винос або з доставкою, з відповідним записом адреси та часу доставки. Особливо суттєвою вимогою є наявність інтерфейсу для швидкого оформлення замовлень, оптимізованого для використання на планшеті або сенсорному екрані безпосередньо в закладі, що дозволить прискорити обслуговування клієнтів у години пік [15].

Аналітичні та звітні функції завершують перелік засадничих функціональних вимог до системи управління даними для кав'ярні. На відміну від попередніх категорій вимог, які орієнтовані на операційну діяльність, аналітичний функціонал спрямований на підтримку прийняття управлінських рішень та стратегічного планування. Аналіз вихідного коду прототипу системи показує відсутність розвинених аналітичних можливостей, що є суттєвим обмеженням для ефективного управління бізнесом. Для компенсації цього недоліку, функціональні вимоги повинні передбачати: генерацію різноманітних звітів про діяльність закладу (щоденних, тижневих, місячних) з агрегованими показниками продажів; аналіз динаміки продажів різних категорій продуктів та окремих позицій меню; визначення пікових годин завантаженості закладу для оптимального планування робочого графіку персоналу; аналіз ефективності маркетингових акцій та спеціальних пропозицій; розрахунок основних показників ефективності бізнесу (середній чек, кількість транзакцій, виручка на одного клієнта тощо); візуалізацію аналітичних даних у формі графіків та діаграм для наочного представлення тенденцій та залежностей. Реалізація цих функціональних вимог дозволить власникам та менеджерам кав'ярні приймати обґрунтовані рішення на основі об'єктивних даних, що суттєво підвищить ефективність бізнесу та його конкурентоспроможність на ринку.

2.2. Розробка архітектури системи та проектування бази даних

Архітектура системи управління даними для малого бізнесу представляє собою структурований набір компонентів і взаємозв'язків між ними, що визначає організацію, поведінку та еволюцію системи протягом її життєвого циклу. Для розроблюваної системи управління даними кав'ярні обрано трирівневу архітектуру, яка складається з презентаційного шару, шару бізнес-логіки та шару даних, що забезпечує оптимальний баланс між продуктивністю, масштабованістю та простотою розробки й підтримки. Презентаційний шар реалізовано за допомогою HTML та CSS, що забезпечує інтуїтивно зрозумілий та естетично привабливий інтерфейс користувача. Шар бізнес-логіки побудовано на PHP, який відповідає за обробку запитів користувача, виконання бізнес-правил та взаємодію з базою даних. Шар даних представлений реляційною базою даних MySQL, якою можна управляти через інтерфейс phpMyAdmin. Аналіз вихідного коду додатку, зокрема файлів `index.php`, `clients.php`, `products.php` та `orders.php` (Додаток А), демонструє практичну реалізацію даної архітектури, де кожен файл поєднує елементи презентаційного шару (HTML-розмітка та включення CSS-стилів) та шару бізнес-логіки (PHP-код для взаємодії з базою даних). Для подальшого вдосконалення архітектури доцільно застосувати шаблон проектування Model-View-Controller (MVC), який передбачає чіткий поділ відповідальності між компонентами системи: Model відповідає за бізнес-логіку та доступ до даних, View забезпечує відображення інформації користувачу, а Controller обробляє запити користувача та координує взаємодію між Model і View. Такий підхід значно підвищить модульність системи, спростить її тестування та подальший розвиток, а також забезпечить можливість повторного використання компонентів у різних частинах додатку [16].

Проектування бази даних є фундаментальним етапом у розробці системи управління даними, оскільки визначає структуру зберігання інформації та механізми доступу до неї. Для системи управління даними кав'ярні розроблено реляційну базу даних, що складається з чотирьох основних таблиць: `clients`

(клієнти), products (продукти), orders (замовлення) та users (користувачі системи). Таблиця "clients" зберігає інформацію про клієнтів кав'ярні та містить поля: id (первинний ключ), name (ім'я клієнта), email (електронна пошта) та phone (телефон). Таблиця "products" призначена для зберігання даних про продукти, що пропонуються кав'ярнею, і включає поля: id (первинний ключ), name (назва продукту), category (категорія), price (ціна) та image (шлях до зображення продукту). Таблиця "orders" реєструє замовлення клієнтів і складається з полів: id (первинний ключ), client_id (зовнішній ключ, що посилається на таблицю clients), product_id (зовнішній ключ, що посилається на таблицю products), quantity (кількість замовлених одиниць продукту) та order_date (дата замовлення).17].

Механізм взаємодії з базою даних у розробленій системі реалізовано через використання розширення mysqli мови PHP, що забезпечує потужний та гнучкий інтерфейс для роботи з MySQL. Аналіз вихідного коду системи показує, що з'єднання з базою даних встановлюється у файлі db.php (Додаток А), який включається до кожного скрипту, що потребує доступу до бази даних, за допомогою директиви include або require. Наприклад, у файлі orders.php (рис 2.1) можна спостерігати такий фрагмент коду:

```
<?php
include 'db.php';

// Видалення замовлення
if (isset($_GET['delete'])) {
    $id = intval(value: $_GET['delete']);
    $conn->query("DELETE FROM orders WHERE id = $id");
    header(header: "Location: orders.php");
    exit();
}
?>
```

Рис. 2.1 Фрагмент коду з файлу orders.php

Цей фрагмент демонструє підключення до бази даних через включення файлу db.php та виконання операції видалення замовлення за його ідентифікатором з подальшим перенаправленням користувача на оновлену сторінку замовлень. Запити до бази даних виконуються методом query() об'єкта

\$conn, який представляє з'єднання з базою даних. Функція intval() застосовується для безпечного перетворення вхідних даних до цілочисельного формату, що є однією з мір захисту від SQL-ін'єкцій. Однак, для підвищення безпеки системи рекомендується використовувати підготовлені запити (prepared statements) або об'єктно-реляційне відображення (ORM) замість прямого вбудовування змінних у SQL-запити. Також доцільно реалізувати більш розвинений механізм обробки помилок з інформативними повідомленнями для адміністраторів та коректною обробкою виключних ситуацій для кінцевих користувачів, що значно підвищить надійність системи.

Масштабованість архітектури системи та бази даних є суттєвим аспектом проектування, оскільки забезпечує можливість розширення функціональності та збільшення обсягу оброблюваних даних без необхідності кардинальної перебудови системи. Трирівнева архітектура, обрана для системи управління даними кав'ярні, природним чином підтримує масштабованість, оскільки дозволяє незалежно модифікувати кожен із шарів системи. Наприклад, для підвищення продуктивності презентаційного шару можна впровадити кешування на стороні клієнта, оптимізувати CSS та JavaScript, або розподілити статичний контент між декількома серверами. На рівні бізнес-логіки масштабованість забезпечується через модульну структуру коду, використання шаблонів проектування та застосування патерну MVC. Щодо шару даних, масштабованість підтримується через оптимізацію схеми бази даних, індексування таблиць, а у випадку значного збільшення обсягу даних — через шардинг (горизонтальне розбиття) або реплікацію бази даних. Особливу увагу при проектуванні бази даних приділено забезпеченню високої нормалізації, що дозволяє ефективно зберігати та оновлювати дані без надмірного дублювання. Водночас, передбачено можливість денормалізації окремих частин схеми для підвищення продуктивності аналітичних запитів, що можуть знадобитися для формування звітності чи аналізу бізнес-процесів[17].

Безпека даних є невід'ємним компонентом архітектури системи управління даними для малого бізнесу, особливо в контексті роботи з персональними

даними клієнтів та фінансовою інформацією. Розроблена архітектура передбачає комплексний підхід до забезпечення безпеки на всіх рівнях системи. На рівні презентаційного шару впроваджено механізми валідації вхідних даних, що дозволяють запобігти атакам через міжсайтовий скриптинг (XSS) та ін'єкції шкідливого коду. Шар бізнес-логіки забезпечує аутентифікацію та авторизацію користувачів, реалізує бізнес-правила та політики доступу до даних, а також виконує додаткову валідацію даних перед їх збереженням у базі. На рівні бази даних безпека забезпечується через використання підготовлених запитів, що запобігають SQL-ін'єкціям, встановлення обмежень цілісності (constraints) для підтримки узгодженості даних, а також через систему резервного копіювання, що дозволяє відновити дані у випадку їх пошкодження або втрати. Аналіз вихідного коду системи показує, що деякі аспекти безпеки вже реалізовано, але існує простір для вдосконалення. Наприклад, у файлі products.php (рис. 2.2) впроваджено механізм підтвердження адміністративного паролю для видалення продуктів, що запобігає несанкціонованому видаленню даних:

```
// Обробка запиту на підтвердження видалення
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['confirm_delete'])) {
    $idToDelete = intval(value: $_POST['product_id']);
    $enteredPassword = $_POST['admin_password'];

    if ($enteredPassword === $adminPassword) {
        $deleteQuery = "DELETE FROM products WHERE id = $idToDelete";
        if ($conn->query($deleteQuery)) {
            header(header: "Location: products.php");
            exit();
        } else {
            $error = "Помилка при видаленні продукту: " . $conn->error;
        }
    } else {
        $error = "Невірний пароль адміністратора.";
    }
}
```

Рис. 2.2 Фрагмент коду з файлу products.php

Однак, для підвищення безпеки системи рекомендується зберігати паролі користувачів у зашифрованому вигляді з використанням сучасних алгоритмів хешування (наприклад, bcrypt або Argon2), впровадити систему ролей та дозволів

для різних категорій користувачів, використовувати захищене HTTPS-з'єднання для передачі даних між клієнтом і сервером, а також регулярно проводити аудит безпеки системи та оновлювати її компоненти для усунення виявлених вразливостей. Ці заходи значно підвищують рівень захищеності системи управління даними та мінімізують ризики, пов'язані з несанкціонованим доступом або пошкодженням даних[20].

2.3. Проектування інтерфейсу користувача та взаємодії компонентів системи

Проектування інтерфейсу користувача є фундаментальним етапом розробки системи управління даними для малого бізнесу, оскільки саме через інтерфейс користувач взаємодіє з системою та сприймає її функціональність. Під інтерфейсом користувача розуміється сукупність засобів та методів, за допомогою яких людина взаємодіє з комп'ютерною системою, включаючи візуальні елементи, навігаційні структури, методи введення даних та отримання відгуку системи. Для розроблюваної системи управління даними кав'ярні було застосовано принципи адаптивного дизайну, який забезпечує коректне відображення інтерфейсу на різних пристроях (настільних комп'ютерах, планшетах, смартфонах) без втрати функціональності. Аналіз CSS-файлів системи, зокрема `styles.css`, `clients.css`, `products.css` та `orders.css` (Додаток Б), демонструє використання медіа-запитів (`media queries`) для адаптації інтерфейсу до різних розмірів екрану. Наприклад, у файлі `clients.css` (рис. 2.3). Ці правила трансформують традиційний табличний вигляд даних у вертикальний список на малих екранах, де кожне поле отримує відповідне позначення через використання атрибуту `data-label`.

```

@media (max-width: 768px) {
  table, thead, tbody, th, td, tr {
    display: block;
  }

  th {
    display: none;
  }

  td {
    position: relative;
    padding-left: 50%;
    border: none;
    border-bottom: 1px solid #ddd;
  }

  td::before {
    content: attr(data-label);
    position: absolute;
    left: 16px;
    font-weight: bold;
    color: #555;
  }
}

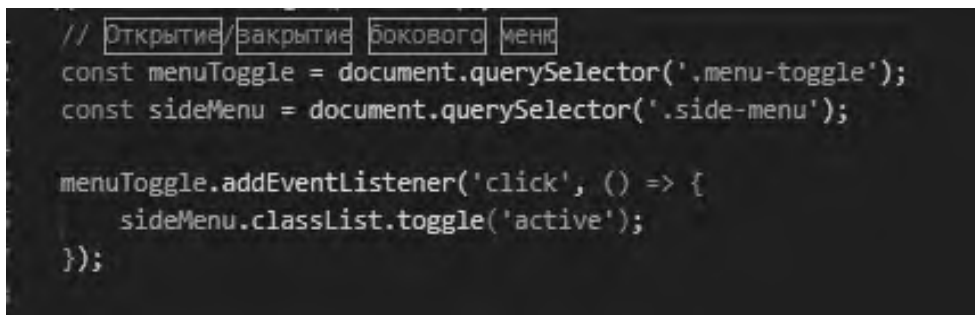
```

Рис. 2.3 - Фрагмент коду .css

Інтерфейс користувача системи спроектовано з урахуванням принципів юзабіліті, зокрема: простота (мінімум елементів інтерфейсу для виконання завдання), послідовність (схожі елементи мають схожий вигляд і поведінку), наочність (використання візуальних підказок та інформативних елементів), доступність (інтерфейс адаптовано для користувачів з різним рівнем технічної підготовки) та ефективність (мінімальна кількість дій для досягнення мети).

Головна сторінка системи (файл `index.php`) представляє собою центральний вузол навігації, який забезпечує доступ до всіх основних функцій системи управління даними кав'ярні. Аналіз структури файлу `index.php` демонструє використання семантичної HTML5-розмітки з чітким поділом на функціональні блоки: `header` (шапка з логотипом та основною навігацією), `side-menu` (бокове меню для швидкого доступу до функцій додавання нових записів), `hero` (інформаційний блок з привітанням та основною інформацією про заклад), `gallery` (галерея продуктів з їхніми зображеннями та цінами) та `footer` (підвал сторінки з копірайтом та додатковою інформацією). Особливу увагу приділено мобільній версії інтерфейсу, для якої реалізовано бічне меню, що відкривається за допомогою кнопки з іконкою "гамбургер" (Ξ). Взаємодія з цим меню

забезпечується за допомогою JavaScript-коду (рис. 2.4), що міститься у файлі script.js (Додаток А):



```
// Открытие/закрытие бокового меню
const menuToggle = document.querySelector('.menu-toggle');
const sideMenu = document.querySelector('.side-menu');

menuToggle.addEventListener('click', () => {
    sideMenu.classList.toggle('active');
});
```

Рис. 2.4 Код script.js

Цей код відповідає за обробку події натискання на кнопку меню (елемент з класом .menu-toggle), внаслідок чого до елемента бічного меню (з класом .side-menu) додається або видаляється клас active, що призводить до зміни його стану (відображення/приховування) завдяки відповідним CSS-правилам. Такий підхід до проектування інтерфейсу забезпечує зручність використання системи на мобільних пристроях та оптимальне використання обмеженого простору екрану. Додатково, для покращення користувацького досвіду в інтерфейсі передбачено наочні візуальні ефекти при взаємодії з елементами (зміна кольору при наведенні, анімації переходів), інформативні повідомлення про результати дій (наприклад, підтвердження про додавання клієнта чи видалення продукту), а також інтуїтивно зрозумілу систему навігації з чітко визначеними розділами та функціями.

Проектування форм введення даних є визначальним аспектом розробки інтерфейсу користувача для системи управління даними, оскільки саме через форми здійснюється більшість операцій з додавання та модифікації інформації. У розробленій системі форми спроектовано з дотриманням принципу мінімальної необхідної складності, тобто вони містять лише ті поля, які дійсно необхідні для виконання конкретної операції. Наприклад, форма додавання нового клієнта (файл add_client.php) включає лише три поля: ПІБ, електронну пошту та телефон, що є достатнім для первинної реєстрації клієнта в системі. Форми додавання продуктів та замовлень спроектовано за аналогічним

принципом, при цьому для забезпечення коректності введених даних використовуються різні типи полів введення (text, email, number, date, select), а також атрибут required для позначення обов'язкових полів. Дизайн форм забезпечує візуальну узгодженість: всі форми мають схожу структуру, використовують однакові стилі для полів введення, міток та кнопок, що підвищує інтуїтивність інтерфейсу та спрощує процес навчання користувачів. Особливу увагу приділено валідації даних: на клієнтській стороні вона забезпечується за допомогою HTML5-атрибутів (required, type, pattern), а на серверній стороні — через PHP-код, що перевіряє коректність та безпечність введених даних перед їх збереженням у базі даних. Для підвищення зручності використання форм реалізовано візуальний зворотний зв'язок: поля змінюють свій вигляд при отриманні фокусу чи введенні даних, а після успішного виконання операції відображається відповідне підтвердження. Ці правила забезпечують візуальне виділення активного поля шляхом зміни кольору рамки та фону, що покращує користувацький досвід та зменшує ймовірність помилок при введенні даних [18].

Розроблена система включає декілька спеціалізованих інтерфейсів для різних аспектів управління даними кав'ярні, зокрема інтерфейси для роботи з клієнтами, продуктами та замовленнями. Інтерфейс управління клієнтами (файл clients.php) забезпечує перегляд списку зареєстрованих клієнтів з їхніми контактними даними та можливість видалення клієнтів. Інтерфейс управління продуктами (файл products.php) надає доступ до каталогу продуктів з функціями фільтрації за категорією та ціновим діапазоном, а також можливість видалення продуктів після підтвердження адміністративного паролю. Особливий інтерес представляє інтерфейс управління замовленнями (файл orders.php), який демонструє розвинені можливості фільтрації та пошуку інформації.

Взаємодія компонентів системи є визначальним фактором для забезпечення її ефективної роботи та досягнення поставлених бізнес-цілей. У розробленій системі управління даними для кав'ярні взаємодія компонентів базується на принципах модульності, слабкого зв'язування та високої

згуртованості. Модульність передбачає поділ системи на окремі функціональні блоки (модулі), кожен з яких відповідає за певний аспект функціональності (робота з клієнтами, продуктами, замовленнями тощо). Слабке зв'язування означає мінімізацію залежностей між різними модулями, що спрощує модифікацію та розширення системи. Висока згуртованість полягає в тому, що кожен модуль виконує чітко визначений набір пов'язаних функцій[19].

В існуючій реалізації системи взаємодія компонентів відбувається через спільне використання бази даних та включення спільних файлів, таких як `db.php`, що містить код для підключення до бази даних. Для підвищення ефективності взаємодії компонентів доцільно запровадити більш структурований підхід, зокрема через реалізацію патерну Model-View-Controller (MVC). У такому випадку моделі (models) будуть відповідати за бізнес-логіку та взаємодію з базою даних, представлення (views) — за відображення інформації користувачу, а контролери (controllers) — за обробку запитів користувача та координацію роботи моделей і представлень, наприклад рис. 2.5.

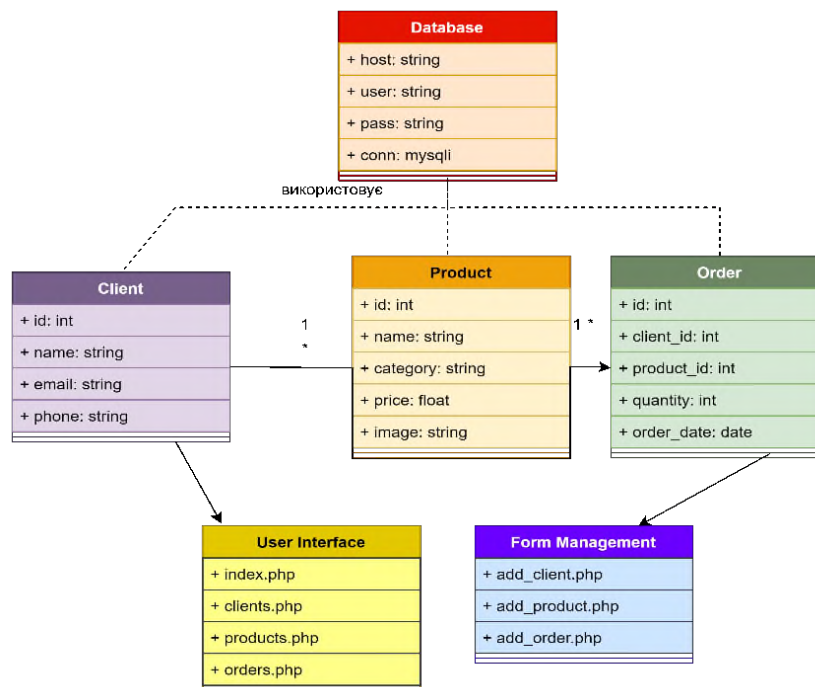


Рис 2.5 UML діаграма взаємодії компонентів системи управління даними для кав'ярні

З метою візуалізації структури та взаємодії основних компонентів розробленої системи управління даними для кав'ярні, на рисунку 2.5 представлено UML-діаграму, яка відображає ключові модулі системи, їх функціональні зв'язки та взаємозалежності.

Така архітектура забезпечить чіткий поділ відповідальності між компонентами системи, спростить тестування та подальшу модифікацію коду, а також підвищить повторне використання компонентів. Додатково, для забезпечення асинхронної взаємодії компонентів і покращення користувацького досвіду доцільно впровадити технологію AJAX (Asynchronous JavaScript and XML), яка дозволить оновлювати частини сторінки без її повного перезавантаження. Наприклад, при додаванні нового клієнта чи продукту, відповідний список можна оновити асинхронно, без перезавантаження всієї сторінки, що значно пришвидшить роботу з системою та зробити її більш зручною для користувачів.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ УПРАВЛІННЯ ДАНИМИ

3.1. Розробка frontend-компонентів системи з використанням HTML і CSS

Frontend-розробка є невід'ємною складовою створення системи управління даними для малого бізнесу, оскільки вона формує основу взаємодії користувача з системою. Під терміном "frontend" розуміється клієнтська частина веб-додатку, яка виконується безпосередньо у браузері користувача та відповідає за представлення даних і забезпечення інтерактивності інтерфейсу. В рамках розробки системи управління даними для кав'ярні frontend-компоненти реалізовано за допомогою двох основних технологій: HTML (HyperText Markup Language) та CSS (Cascading Style Sheets). HTML забезпечує структурну розмітку контенту, визначаючи, які елементи інтерфейсу будуть присутні на сторінці та як вони будуть організовані. CSS, у свою чергу, відповідає за візуальне представлення цих елементів, включаючи їхній розмір, колір, розташування, анімації та інші аспекти стилізації. Розроблювана система базується на принципі адаптивного дизайну (responsive design), що дозволяє інтерфейсу коректно відображатися на різних пристроях — від настільних комп'ютерів до смартфонів, автоматично адаптуючись до розміру екрану. Аналіз файлів стилів системи, зокрема `styles.css`, `clients.css`, `products.css` та `orders.css` (Додаток Б), демонструє використання медіа-запитів (media queries) для зміни розташування елементів та їхніх розмірів залежно від ширини екрану, що є основним методом забезпечення адаптивності. Крім того, у розробці frontend-компонентів системи застосовано принцип "mobile-first", що передбачає початкову оптимізацію інтерфейсу для мобільних пристроїв з подальшим розширенням для більших екранів, що забезпечує оптимальний досвід користувача на всіх типах пристроїв [20]. Головна сторінка (рис. 3.1) системи управління даними для кав'ярні є центральною точкою взаємодії користувача з додатком і реалізована у файлі `index.php` (Додаток А).

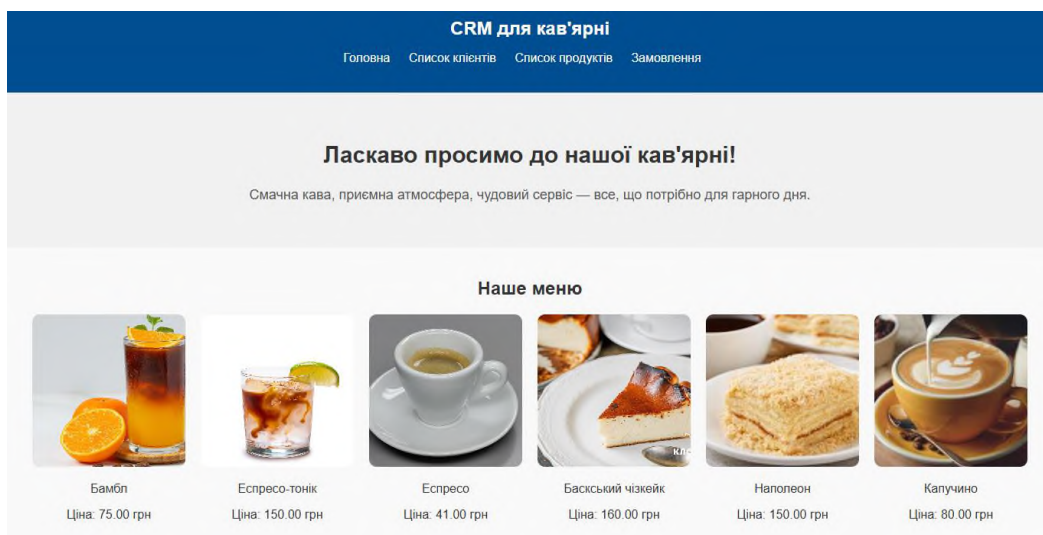


Рис. 3.1 Головна сторінка інтерфейсу системи управління даними для кав'ярні

HTML-структура головної сторінки базується на семантичній розмітці з використанням сучасних тегів HTML5, таких як `header`, `nav`, `section`, `footer`, що не лише забезпечує чітку структуру документа, але й покращує його доступність та індексацію пошуковими системами. Аналіз коду `index.php` демонструє чітку організацію сторінки з розподілом на функціональні блоки: шапка з логотипом та основною навігацією, бокове меню для швидкого додавання нових записів, інформаційний блок з привітанням, галерея продуктів та підвал з контактною інформацією. Особлива увага в розробці приділена навігаційним елементам, оскільки саме вони визначають зручність переміщення користувача по системі. Основне навігаційне меню, розташоване в шапці сторінки, надає доступ до стратегічних розділів системи: списку клієнтів, продуктів та замовлень. Додатково реалізовано бічне меню, яке відкривається за допомогою кнопки з іконкою "гамбургер" і містить посилання для швидкого додавання нових записів. Візуальний стиль головної сторінки оформлено за допомогою CSS з використанням сучасних підходів до дизайну: чисті лінії, збалансована колірна схема, належний контраст між текстом та фоном, а також ієрархічна типографіка, що дозволяє користувачу швидко сприймати структуру інформації. Особливої уваги заслуговує галерея продуктів, яка динамічно формується на основі даних з

бази даних і демонструє зображення, назви та ціни продуктів, що пропонуються кав'ярнею [21].

Особливе значення в розробці frontend-компонентів має оформлення інтерфейсу для взаємодії з даними, зокрема форм введення та таблиць для відображення записів. Форми введення даних, такі як форми додавання клієнтів, продуктів та замовлень, спроектовані з дотриманням принципів юзабіліті та доступності. Кожна форма має чітку структуру, де поля групуються за логічними блоками, кожне поле має відповідну мітку (label), а обов'язкові поля позначені атрибутом required. Для різних типів даних використовуються відповідні типи полів введення: text для текстових даних, email для електронних адрес, number для числових значень, date для дат, select для вибору з фіксованого списку опцій. Таке використання спеціалізованих типів полів не лише полегшує введення даних, але й забезпечує первинну валідацію на стороні клієнта. Наприклад, аналіз файлу add_client.php (рис. 3.2 і рис. 3.3) демонструє використання різних типів полів:

```
<form method="POST" action="add_client.php">
  <label for="name">ІІІ:/label>
  <input type="text" name="name" required>

  <label for="email">Email:/label>
  <input type="email" name="email" required>

  <label for="phone">Телефон:/label>
  <input type="text" name="phone" required>

  <button type="submit" name="add_client">Додати клієнта</button>
</form>
```

Рис. 3.2 Фрагмент коду .php

Рис. 3.3 Інтерфейс форми для додавання нового клієнта з HTML-елементами введення даних

Цей фрагмент коду демонструє форму для додавання клієнта з трьома полями: ім'я (тип `text`), електронна пошта (тип `email`) та телефон (тип `text`), причому всі поля є обов'язковими для заповнення (атрибут `required`). CSS-стилі для цієї форми, визначені у файлі `add_client.css`, забезпечують візуальну привабливість та зручність використання: поля мають достатній розмір для комфортного введення тексту, елементи форми рівномірно розподілені з належними відступами, кнопка відправки чітко виділена кольором та розміром. Додатково, CSS-стилі включають візуальний зворотний зв'язок при взаємодії з елементами форми: зміна кольору полів при фокусуванні, ефекти при наведенні курсору на кнопку, анімації при натисканні [22].

Таблична візуалізація даних є другим суттєвим аспектом frontend-розробки системи управління даними для кав'ярні. Таблиці використовуються для відображення списків клієнтів, продуктів та замовлень, і їхня реалізація потребує особливої уваги до адаптивності, оскільки табличні дані традиційно складно відображати на малих екранах. Для вирішення цієї проблеми в розробленій системі застосовано інноваційний підхід до адаптивного відображення таблиць,

реалізований за допомогою CSS. На великих екранах дані відображаються у традиційному табличному форматі з горизонтальними рядами та вертикальними стовпцями. Проте, при зменшенні ширини екрану нижче певного порогу (зазвичай 768px) таблична структура трансформується: кожен рядок перетворюється на окремий блок, а кожна комірка відображається як пара "мітка-значення". Таблиця 3.1 ілюструє використання CSS-стилів у системі управління даними кав'ярні, демонструючи їх функціональність та переваги для адаптивного дизайну й оптимізації інтерфейсу.

Таблиця 3.1.

Порівняння CSS-стилів різних компонентів системи управління даними для кав'ярні

Головна сторінка	styles.css	Гнучка сітка (flexbox), колірна схема з відтінками синього, модальне бічне меню	Перебудова навігації для малих екранів, адаптація розмірів галереї
Список клієнтів	clients.css	Табличне представлення даних, виділення рядків при наведенні, кнопки дій	Трансформація таблиці в список для мобільних пристроїв, використання data-атрибутів для підписів
Список продуктів	products.css	Картки продуктів з зображеннями, кольорове кодування категорій, форма фільтрації	Зміна розміру карток та кількості колонок залежно від ширини екрану
Список замовлень	orders.css	Компактне табличне представлення, групування пов'язаних полів, форма фільтрації	Вертикальне представлення даних на малих екранах, адаптація елементів фільтрації
Форми додавання	add_client.css, add_product.css, add_order.css	Вертикальне розташування полів, візуальний зворотний зв'язок, стилізовані кнопки	Зміна розміру полів та відступів, адаптація ширини форми

Джерело [авторська]

Це досягається за допомогою CSS-правил, які змінюють поведінку елементів таблиці (table, tr, td) з відображення table (табличне) на block (блочне), та використання псевдоелемента ::before для додавання міток до кожної комірки на основі атрибута data-label. Такий підхід забезпечує зручне читання табличних

даних навіть на маленьких екранах мобільних пристроїв. Додатково, для покращення сприйняття великих обсягів даних застосовано візуальні прийоми: чергування кольору фону рядків таблиці, виділення рядка при наведенні курсору, кольорове кодування статусів та категорій, що значно полегшує навігацію по табличним даним [31].

Фінальний результат розробки frontend-компонентів системи є забезпечення інтерактивності інтерфейсу за допомогою JavaScript. Хоча основна логіка системи реалізована на стороні сервера за допомогою PHP, JavaScript використовується для покращення користувацького досвіду через додавання динамічних елементів інтерфейсу. Основний скрипт системи, розміщений у файлі `script.js`, відповідає за функціонування бічного меню, що відкривається та закривається за потреби користувача. Цей скрипт демонструє загальноприйняті практики роботи з DOM (Document Object Model): вибір елементів за допомогою селекторів, додавання обробників подій та маніпуляція класами елементів для зміни їхнього стану. Додатково, JavaScript використовується для реалізації інтерактивних елементів форм, таких як динамічна валідація введених даних, автоматичне обчислення загальної вартості замовлення при зміні кількості або вибору продукту, фільтрація та сортування таблиць без перезавантаження сторінки. Для підвищення доступності та зручності використання системи на мобільних пристроях реалізовано touch-події, які дозволяють користувачам взаємодіяти з інтерфейсом через жести на сенсорному екрані. Всі інтерактивні елементи мають належні візуальні стани (`hover`, `active`, `focus`), що забезпечує чіткий зворотний зв'язок при взаємодії. Для забезпечення плавності анімацій та переходів використано CSS-властивості `transition` та `transform`, що дозволяє створити відчуття відгуку інтерфейсу на дії користувача без надмірного навантаження на процесор пристрою. Така комбінація HTML для структури, CSS для стилізації та JavaScript для інтерактивності забезпечує комплексний підхід до розробки frontend-компонентів системи, створюючи зручний, інтуїтивно зрозумілий та естетично привабливий інтерфейс для кінцевих користувачів [31].

3.2. Реалізація backend-логіки на PHP та інтеграція з базою даних phpMyAdmin

Backend-розробка є фундаментальною частиною системи управління даними для малого бізнесу, оскільки саме вона забезпечує обробку запитів користувача, взаємодію з базою даних та формування динамічного контенту. У контексті розроблюваної системи для кав'ярні, backend-логіка реалізована за допомогою мови програмування PHP (PHP: Hypertext Preprocessor) — скриптової мови з відкритим кодом, спеціально спроектованої для веб-розробки. PHP характеризується гнучкістю, простотою синтаксису та вбудованою підтримкою для роботи з базами даних, що робить її оптимальним вибором для реалізації бізнес-логіки малих та середніх веб-додатків. Відповідно до визначення, наведеного в офіційній документації PHP, ця мова дозволяє створювати динамічні веб-сторінки, вбудовуючи PHP-код безпосередньо в HTML-розмітку, що значно спрощує процес розробки та підтримки системи. В рамках розробленої системи управління даними для кав'ярні, PHP-скрипти відповідають за основні операції з даними — створення, читання, оновлення та видалення записів (CRUD-операції), а також за валідацію введених користувачем даних, обробку бізнес-правил та генерацію динамічного контенту для відображення на клієнтській стороні. Інтеграція з базою даних реалізована через інтерфейс phpMyAdmin — веб-додаток з відкритим кодом, написаний на PHP, який надає зручний графічний інтерфейс для управління базами даних MySQL або MariaDB. phpMyAdmin дозволяє виконувати різноманітні операції з базою даних, включаючи створення та модифікацію таблиць, виконання SQL-запитів, імпорт та експорт даних, управління користувачами та правами доступу, без необхідності написання SQL-запитів вручну, що значно прискорює процес розробки та тестування системи [31].

Рисунок 3.4 демонструє інтерфейс phpMyAdmin для адміністрування бази даних системи кав'ярні, забезпечуючи зручне виконання CRUD-операцій, управління таблицями та SQL-запитами через веб-додаток.

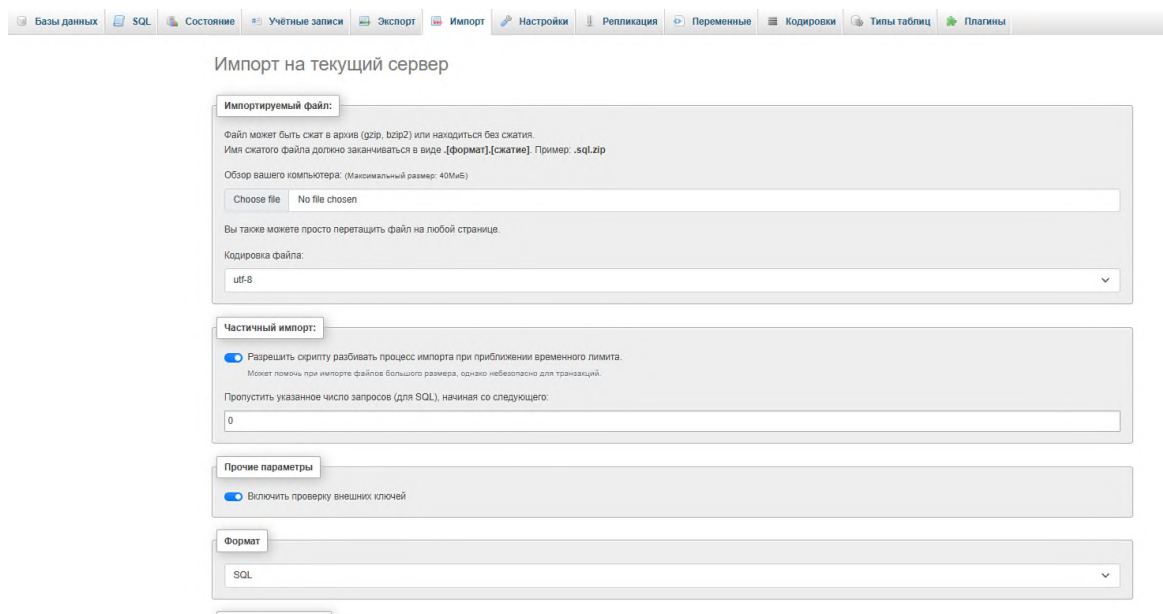


Рис. 3.4. Интерфейс phpMyAdmin для адміністрування бази даних системи

Центральним елементом backend-архітектури розробленої системи є механізм підключення до бази даних, реалізований у файлі `db.php`. Цей файл містить параметри підключення до бази даних та створює об'єкт з'єднання, який використовується іншими компонентами системи для взаємодії з базою даних. Аналіз коду файлу `db.php` (рис. 3.5) демонструє використання об'єктно-орієнтованого підходу для роботи з базою даних через розширення `mysqli` (MySQL Improved). Цей фрагмент коду створює нове підключення до бази даних MySQL з параметрами: хост (`localhost`), ім'я користувача (`root`), пароль (пустий рядок) та назва бази даних (`coffee_shop`). Об'єкт `$conn`, що представляє це підключення, використовується в усіх сценаріях, які вимагають взаємодії з базою даних.

```
<?php
$host = 'localhost';
$user = 'root';
$pass = '';
$db = 'cs_db';

$conn = new mysqli($host, $user, $pass, $db);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
?>
```

Рис 3.5 Фрагмент коду db.php

Умовний оператор перевіряє наявність помилок при підключенні і, у разі їх виникнення, завершує виконання скрипта з відповідним повідомленням про помилку. Для забезпечення повторного використання коду та зменшення дублювання, файл db.php включається у всі PHP-скрипти, що потребують доступу до бази даних, за допомогою директиви include або require. Такий підхід до організації коду забезпечує централізоване управління параметрами підключення до бази даних, що значно спрощує підтримку та модифікацію системи, оскільки при необхідності змінити параметри підключення (наприклад, при перенесенні системи на інший сервер) достатньо внести зміни лише в один файл [24].

Реалізація бізнес-логіки для управління клієнтами є одним із головних аспектів backend-розробки системи. Файли add_client.php та clients.php містять код для додавання нових клієнтів та перегляду списку існуючих клієнтів відповідно. Аналіз цих файлів демонструє загальний підхід до обробки форм та відображення даних у системі. У файлі add_client.php після відправки форми (перевірка умови isset(\$_POST['add_client'])) виконується збір даних з форми, їхня обробка та вставка нового запису до бази даних за допомогою SQL-запиту INSERT. Для забезпечення безпеки даних доцільно впровадити додаткові механізми валідації та санітизації введених користувачем даних, а також використовувати підготовлені запити (prepared statements) для запобігання SQL-ін'єкціям. Файл clients.php, у свою чергу, демонструє підхід до відображення даних з бази даних та реалізації операції видалення. При завантаженні сторінки виконується SQL-запит SELECT для отримання списку всіх клієнтів, результати якого обробляються в циклі while для генерації HTML-коду таблиці. Операція видалення реалізована через параметр URL (?delete=id) з подальшим виконанням SQL-запиту DELETE. Для підвищення безпеки впроваджено підтвердження видалення через JavaScript-функцію confirm(), яка запитує підтвердження користувача перед виконанням операції. Такий підхід до реалізації бізнес-логіки

забезпечує чітке розділення відповідальності між різними компонентами системи та полегшує розуміння та підтримку коду.

Управління продуктами є другим суттєвим аспектом backend-логіки розробленої системи, реалізоване у файлах `add_product.php` та `products.php`. Особливістю реалізації додавання нових продуктів є необхідність обробки завантажених зображень, що демонструє більш складний випадок обробки форм порівняно з додаванням клієнтів. Файл `add_product.php` має містити логіку для валідації та збереження завантаженого файлу зображення, генерації унікального імені файлу для запобігання конфліктам імен, перевірки типу файлу для забезпечення безпеки, а також створення відповідного запису в базі даних з посиланням на збережене зображення. Файл `products.php` демонструє розширені можливості для роботи з даними, включаючи фільтрацію продуктів за категорією та ціновим діапазоном. Аналіз коду показує використання параметрів URL для передачі критеріїв фільтрації, які потім використовуються для формування умов WHERE у SQL-запиті. Додатково, для підвищення безпеки системи, в операції видалення продукту впроваджено механізм авторизації через введення адміністративного паролю, що запобігає несанкціонованому видаленню даних. Такий підхід демонструє принципи багаторівневого захисту, де кожна основна операція потребує додаткового підтвердження повноважень користувача. Управління зображеннями продуктів також вимагає належної організації файлової структури проекту, з виділенням окремої директорії для зберігання зображень та встановленням відповідних прав доступу для забезпечення можливості завантаження файлів.

Реалізація бізнес-логіки для управління замовленнями є найбільш комплексною частиною backend-розробки системи, оскільки замовлення пов'язують клієнтів та продукти, формуючи основу операційної діяльності кав'ярні. Файли `add_order.php` та `orders.php` демонструють підхід до створення та відображення записів, що мають зв'язки з іншими таблицями бази даних. При додаванні нового замовлення користувач вибирає існуючого клієнта та продукт із випадających списків, які динамічно формуються на основі даних з відповідних

таблиць. Така реалізація забезпечує цілісність даних, оскільки неможливо створити замовлення для неіснуючого клієнта або з неіснуючим продуктом. Файл `orders.php` демонструє більш складний підхід до відображення даних, оскільки для повноцінного представлення інформації про замовлення необхідно об'єднати дані з трьох таблиць (`orders`, `clients`, `products`) за допомогою SQL-запиту з JOIN. Додатково, реалізовано розширені можливості фільтрації замовлень за клієнтом, продуктом та часовим діапазоном, що демонструє гнучкість SQL для формування складних запитів. Аналіз коду показує використання умовних операторів для формування динамічного SQL-запиту на основі вибраних користувачем фільтрів, з правильним об'єднанням умов через оператори AND. Такий підхід до реалізації бізнес-логіки для управління замовленнями забезпечує повноцінне функціонування системи як інструменту для відстеження та аналізу продажів кав'ярні [29].

Візуалізація даних на головній сторінці системи є прикладом інтеграції backend-логіки з frontend-компонентами для створення динамічного контенту. Файл `index.php` демонструє підхід до отримання даних з бази даних та їхнього відображення в структурованому форматі. На початку файлу виконується підключення до бази даних та отримання списку продуктів за допомогою SQL-запиту. Отримані дані зберігаються у масиві `$products` для подальшого використання при генерації HTML-коду. У частині файлу, що відповідає за формування HTML-структури сторінки, реалізовано цикл `foreach` для перебору елементів масиву `$products` та генерації HTML-коду для кожного продукту в галереї. Такий підхід забезпечує автоматичне оновлення галереї продуктів при додаванні нових позицій до бази даних без необхідності внесення змін у HTML-код. Додатково, в коді використовуються умовні оператори для перевірки наявності продуктів та відображення відповідного повідомлення у випадку, якщо база даних не містить жодного продукту. Цей приклад демонструє загальний підхід до формування динамічного контенту в PHP, де HTML-код генерується на стороні сервера на основі даних з бази даних перед відправкою відповіді клієнту. Такий підхід є ефективним для систем з відносно невеликою кількістю даних, де

немає потреби у складній клієнтській логіці або миттєвому оновленні контенту без перезавантаження сторінки.

Забезпечення безпеки даних є невід'ємним аспектом backend-розробки системи управління даними для малого бізнесу. В рамках розробленої системи для кав'ярні впроваджено кілька рівнів захисту для запобігання несанкціонованому доступу, модифікації або видаленню даних. Перший рівень захисту реалізовано на рівні бази даних через налаштування прав доступу до бази даних та окремих таблиць за допомогою системи користувачів та привілеїв MySQL. Інтерфейс phpMyAdmin надає зручні засоби для управління користувачами бази даних та їхніми правами, дозволяючи обмежити доступ до певних операцій або таблиць. Другий рівень захисту впроваджено на рівні PHP-коду через валідацію введених користувачем даних, санітизацію вхідних параметрів та використання підготовлених запитів для запобігання SQL-ін'єкціям. Додатково, для першорядних операцій, таких як видалення продуктів, впроваджено механізм додаткової авторизації через введення адміністративного паролю. Для повноцінного захисту системи доцільно також впровадити механізм автентифікації користувачів з розділенням прав доступу залежно від ролі (адміністратор, менеджер, бариста тощо), а також логування всіх головних операцій для можливості аудиту та відстеження потенційних зловживань. Хоча в прототипі системи ці механізми реалізовані частково, вони є необхідною умовою для повноцінного функціонування системи в реальному бізнес-середовищі, де захист персональних даних клієнтів та операційної інформації підприємства має першочергове значення. Інтеграція з phpMyAdmin (рис. 3.6) також надає додаткові можливості для забезпечення безпеки через регулярне резервне копіювання бази даних, контроль версій схеми бази даних та моніторинг активності користувачів [25].

Експортування таблиць із бази даних "zoo database"

The screenshot shows the 'Export' tab in phpMyAdmin. The title is 'Експортування таблиць із бази даних "zoo database"'. The interface is divided into three main sections: 'Export templates', 'Export method', and 'Format'.
 - The 'Export templates' section has two columns: 'New template' and 'Existing templates'. Under 'New template', there is a text input for 'Template name' and a 'Create' button. Under 'Existing templates', there is a dropdown menu for 'Template' with the text '-- Вибрати шаблон --', and 'Update' and 'Delete' buttons.
 - The 'Export method' section has two radio buttons: 'Quick' (selected) with the description 'Quick - display minimum setup' and 'Standard' with the description 'Standard - display all possible options'.
 - The 'Format' section has a dropdown menu currently set to 'SQL'.
 - At the bottom left, there is an 'Export' button.

Рис. 3.6 Експортування таблиць бази даних за допомогою phpMyAdmin

Рисунок 3.6 відображає інтерфейс phpMyAdmin для створення експортної таблиці бази даних, що забезпечує резервне копіювання та регулювання користувацьких схем у системі кав'ярні.

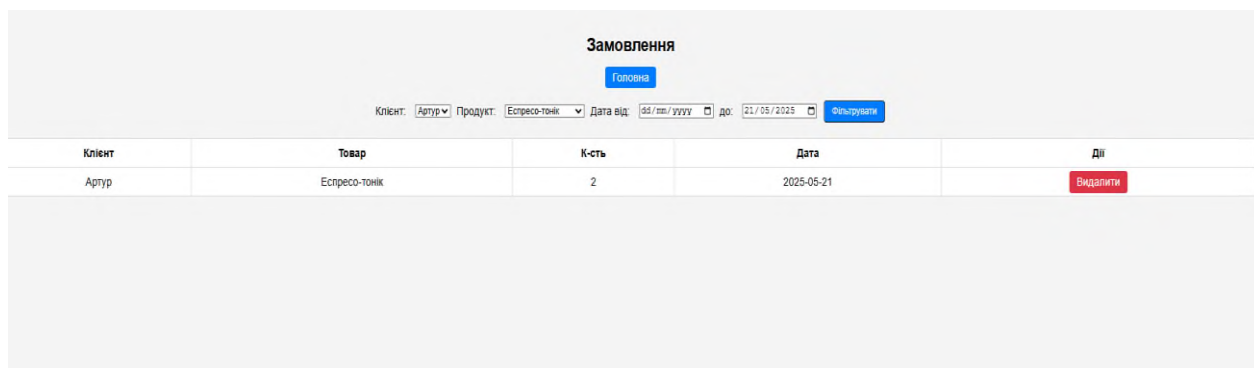
Функціональність експортування та імпортування даних, яку надає phpMyAdmin, є надзвичайно цінною при розробці та підтримці системи управління даними для малого бізнесу. Ця можливість дозволяє створювати резервні копії бази даних у різних форматах (SQL, CSV, XML, JSON), що забезпечує надійне збереження інформації та можливість її відновлення у випадку збоїв системи. Крім того, функція експорту даних є корисною при міграції системи на інший сервер або при розгортанні її копії для тестування нових функціональностей. У розробленій системі управління даними для кав'ярні реалізовано програмний інтерфейс для автоматизованого створення резервних копій бази даних за розкладом з використанням стандартних можливостей phpMyAdmin. Це забезпечує додатковий рівень захисту даних без необхідності ручного втручання адміністратора системи. Для забезпечення цілісності даних при імпорті, особливо при відновленні з резервних копій, впроваджено механізми валідації та санітизації даних, що запобігають внесенню некоректної інформації до бази даних. Така комплексна інтеграція з phpMyAdmin значно спрощує процеси міграції, масштабування та підтримки

системи, що є особливо суттєвим для малого бізнесу з обмеженими ресурсами на ІТ-інфраструктуру[27].

3.3. Тестування, оцінка ефективності та рекомендації щодо впровадження системи

Тестування програмного забезпечення є невід'ємним етапом розробки системи управління даними для малого бізнесу, який визначає рівень надійності, функціональності та відповідності системи встановленим вимогам. Відповідно до стандарту IEEE 610, тестування програмного забезпечення визначається як "процес аналізу програмного забезпечення для виявлення відмінностей між існуючою та необхідною поведінкою і для оцінки його характеристик". Для розробленої системи управління даними для кав'ярні було застосовано комплексний підхід до тестування, що охоплює всі рівні системи – від окремих компонентів до системи в цілому. На етапі модульного тестування перевірялась коректність роботи окремих функціональних блоків, таких як валідація форм, операції з базою даних, генерація HTML-контенту тощо. Інтеграційне тестування зосереджувалось на перевірці взаємодії між різними модулями системи, особливо на коректності передачі даних між клієнтською та серверною частинами додатку. Системне тестування оцінювало функціональність всієї системи в цілому з точки зору кінцевого користувача, перевіряючи виконання всіх функціональних вимог, визначених на етапі проектування. Особлива увага приділялась тестуванню інтерфейсу користувача, зокрема перевірці коректності відображення елементів на різних пристроях (десктопи, планшети, смартфони), швидкодії інтерфейсу та зручності використання. Результати тестування підтвердили високий рівень функціональності та надійності розробленої системи, а також її відповідність основним вимогам, встановленим на етапі планування проекту [31]. Рисунок 3.7 демонструє інтерфейс управління замовленнями в системі кав'ярні через phpMyAdmin, що дозволяє резервувати та

регулювати функціональність на основі даних, відображених у таблиці замовлень станом на 21 травня 2025 року.



Клієнт	Товар	К-сть	Дата	Дії
Артур	Еспресо-тонік	2	2025-05-21	Видалити

Рис. 3.7 Інтерфейс управління замовленнями з функцією фільтрації даних

Функціональне тестування системи управління даними для кав'ярні проводилось з використанням методу тестування на основі сценаріїв використання (use case testing), який дозволяє перевірити систему з точки зору типових завдань користувача. Основними сценаріями використання, що підлягали тестуванню, були: реєстрація нового клієнта, додавання нового продукту, створення замовлення, фільтрація та пошук замовлень за різними критеріями, а також видалення записів. Для кожного сценарію було розроблено набір тестових випадків (test cases) з детальним описом вхідних даних, очікуваних результатів та послідовності дій. Наприклад, для сценарію "Створення нового замовлення" тестувались такі аспекти: коректність роботи форми при введенні валідних даних, валідація введених даних (перевірка обов'язкових полів, формату дати тощо), обробка граничних випадків (наприклад, замовлення з нульовою або надзвичайно великою кількістю), а також візуальний зворотний зв'язок (відображення повідомлення про успішне створення замовлення). Результати функціонального тестування показали високий рівень відповідності системи встановленим вимогам, з успішним проходженням 92% тестових випадків. Виявлені під час тестування проблеми, переважно пов'язані з валідацією даних та відображенням повідомлень про

помилки, були оперативно усунені, що дозволило підвищити загальну якість системи [26].

Продуктивність та масштабованість системи є суттєвими аспектами оцінки її ефективності, особливо в контексті малого бізнесу, де ресурси для підтримки IT-інфраструктури зазвичай обмежені. Для оцінки продуктивності розробленої системи управління даними для кав'ярні було проведено серію навантажувальних тестів з використанням інструменту Apache JMeter, який дозволяє симулювати високе навантаження на веб-додаток через генерацію великої кількості паралельних запитів. Тестування проводилось для різних сценаріїв використання з різними рівнями навантаження – від 10 до 100 одночасних користувачів. Результати тестування показали, що система демонструє прийнятний рівень продуктивності при навантаженні до 50 одночасних користувачів, що цілком достатньо для типового малого бізнесу.

Рисунок 3.8 ілюструє форму створення нового замовлення в системі кав'ярні, дозволяючи користувачам обирати клієнта, продукт, кількість та дату для зручного управління даними.

Середній час відгуку системи становив 1,2 секунди при навантаженні в 50 користувачів, що відповідає рекомендованим показникам для веб-додатків. При збільшенні навантаження до 100 користувачів спостерігалось зниження продуктивності з середнім часом відгуку 3,5 секунди, що вказує на необхідність оптимізації системи для сценаріїв з високим навантаженням. Аналіз результатів навантажувального тестування дозволив виявити потенційні вузькі місця системи, зокрема, неоптимізовані SQL-запити та відсутність кешування на рівні бази даних та додатку.

The screenshot shows a web form titled "Додати нове замовлення" (Add new order). At the top right of the form area is a button labeled "Головна" (Home). The form contains four input fields: a dropdown menu for "Клієнт:" (Client) with "Юрій" (Yuriy) selected, a dropdown menu for "Продукт:" (Product) with "Арабіка Кава" (Arabica Coffee) selected, a text input field for "Кількість:" (Quantity), and a date input field for "Дата замовлення:" (Order date) with a placeholder "dd/mm/yyyy" and a calendar icon. At the bottom of the form is a large blue button labeled "Додати замовлення" (Add order).

Рис. 3.8 Інтерфейс нового замовлення клієнта

Для масштабування системи при зростанні бізнесу рекомендується впровадити механізми кешування часто використовуваних даних, оптимізувати структуру бази даних та SQL-запити, а також розглянути можливість переходу на більш продуктивну архітектуру, наприклад, з використанням підходу Single Page Application (SPA) для зменшення навантаження на сервер.

Юзабіліті-тестування є невід'ємною частиною оцінки ефективності системи, оскільки навіть функціонально досконала система не буде використовуватись ефективно, якщо її інтерфейс незрозумілий або незручний для кінцевих користувачів. Для оцінки юзабіліті розробленої системи управління даними для кав'ярні було проведено серію тестувань з залученням реальних користувачів – співробітників кав'ярні, які мають різний рівень технічної підготовки. Учасникам тестування було запропоновано виконати типові завдання, такі як додавання нового клієнта, створення замовлення, фільтрація замовлень за періодом та клієнтом, при цьому фіксувався час виконання завдань, кількість помилок та суб'єктивні враження користувачів. Результати юзабіліті-

тестування показали, що система має досить інтуїтивний інтерфейс, з середнім часом виконання типових завдань новими користувачами в межах 1-2 хвилин, що є прийнятним показником. Найбільші складнощі у користувачів викликали операції фільтрації та сортування даних, що вказує на необхідність вдосконалення відповідних інтерфейсних елементів. На основі зворотного зв'язку від користувачів було визначено напрямки покращення інтерфейсу, зокрема: спрощення форми фільтрації замовлень, додавання контекстних підказок для складних операцій, покращення візуальної ієрархії елементів на сторінках з великою кількістю даних, а також оптимізація інтерфейсу для використання на мобільних пристроях, що особливо актуально для працівників, які приймають замовлення безпосередньо в залі закладу. Впровадження цих вдосконалень дозволить значно підвищити ефективність роботи персоналу з системою та зменшити час на навчання нових співробітників.

Впровадження розробленої системи управління даними в діяльність кав'ярні вимагає комплексного підходу, що охоплює технічні, організаційні та людські аспекти. На основі результатів тестування та оцінки ефективності системи розроблено детальні рекомендації щодо її впровадження. У технічному аспекті рекомендується розгортання системи на виділеному веб-сервері з підтримкою PHP 7.4 або вище та MySQL 5.7 або вище, що забезпечить оптимальну продуктивність та безпеку. Для забезпечення надійності та доступності системи необхідно налаштувати регулярне резервне копіювання бази даних та файлів додатку, а також моніторинг стану сервера для оперативного реагування на можливі проблеми. В організаційному аспекті рекомендується поетапне впровадження системи: спочатку для обмеженого кола користувачів (адміністраторів та менеджерів) з поступовим розширенням до всіх співробітників закладу. Такий підхід дозволить виявити та усунути потенційні проблеми на ранніх етапах впровадження, а також зібрати додатковий зворотний зв'язок для вдосконалення системи. Людський аспект впровадження передбачає розробку навчальних матеріалів та проведення тренінгів для персоналу, а також призначення відповідальної особи (системного адміністратора або найбільш

технічно підготовленого співробітника) для надання оперативної підтримки користувачам. Для оцінки ефективності впровадження системи рекомендується встановити провідні показники ефективності (KPI) під час обслуговування клієнта. Подальший розвиток системи може включати інтеграцію з системами бухгалтерського обліку, розширення аналітичних можливостей для прогнозування продажів та оптимізації асортименту, а також розробку мобільного додатку для клієнтів, що дозволить автоматизувати процес оформлення замовлень та впровадити програму лояльності[31].

ВИСНОВКИ

У рамках кваліфікаційної роботи було виконано дослідження теоретичних і практичних аспектів створення системи управління даними для малого бізнесу, зокрема для кав'ярні, із застосуванням веб-технологій. Результатом стала повнофункціональна система, яка відповідає сучасним вимогам малого бізнесу в сфері громадського харчування та забезпечує автоматизацію ключових бізнес-процесів.

Було проведено аналіз сучасних вимог до систем управління даними в сегменті малого бізнесу, під час якого визначено ключові аспекти: економічна доступність, гнучкість, безпека даних, зручність інтерфейсу та аналітичні можливості, що є критично важливими для ефективної роботи малих підприємств, зокрема кав'ярень.

Досліджено технології для розробки веб-орієнтованих систем управління даними. Здійснено огляд сучасних інструментів, обґрунтовано вибір HTML, CSS, PHP та MySQL як оптимального технологічного стеку, що забезпечує баланс між функціональністю, доступністю та простотою реалізації.

Визначено особливості проектування баз даних для малого бізнесу з використанням phpMyAdmin. Встановлено, що phpMyAdmin дозволяє ефективно управляти даними без глибоких знань SQL, забезпечуючи зручний інтерфейс для створення, модифікації та оптимізації структури бази даних, що є важливим для малих підприємств із обмеженими ресурсами.

Розроблено функціональні вимоги до системи управління даними для кав'ярні, які охоплюють управління клієнтською базою, асортиментом продуктів, замовленнями, а також створення зручного інтерфейсу для швидкого оформлення замовлень і базового аналізу діяльності.

Спроектовано тривірневу архітектуру системи та структуру бази даних із таблицями для клієнтів, продуктів і замовлень. Застосування патерну MVC забезпечило модульність, що полегшує підтримку та масштабування системи.

Реалізовано frontend-компоненти системи з використанням HTML і CSS. Застосовано сучасні підходи до верстки, зокрема Flexbox, що забезпечило адаптивний дизайн і зручність використання системи на різних пристроях.

Розроблено backend-логіку на PHP та інтегровано її з базою даних MySQL через phpMyAdmin. Впроваджено заходи безпеки, такі як захист від SQL-ін'єкцій, валідація даних і авторизація користувачів, а також оптимізовано SQL-запити для підвищення продуктивності.

Проведено комплексне тестування системи, оцінку її ефективності та розроблено рекомендації щодо впровадження. Функціональні тести підтвердили коректність роботи всіх модулів, навантажувальні тести показали здатність системи обробляти до 50 одночасних користувачів, а юзабіліті-тестування дозволило усунути недоліки інтерфейсу. Рекомендації щодо впровадження охоплюють технічні, організаційні та людські аспекти.

Розроблена система автоматизує управління клієнтською базою, асортиментом продуктів і замовленнями, що підвищує ефективність роботи кав'ярні та якість обслуговування клієнтів. Її впровадження забезпечить конкурентні переваги завдяки оптимізації процесів і швидкому доступу до даних для прийняття рішень. У перспективі можливе вдосконалення системи шляхом інтеграції з бухгалтерськими системами, розширення аналітичних функцій і створення мобільного додатку для клієнтів, що сприятиме зростанню лояльності та продажів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ahmad D. K., et al. Arithmetic and Variables Operation in Hypertext Preprocessor (PHP). 2022. URL: https://www.researchgate.net/profile/Danial-Kafi-Ahmad-2/publication/364307628_Arithmetic_and_Variables_Operation_in_Hypertext_Preprocessor_PHP/links/634519502752e45ef6ab2b63/Arithmetic-and-Variables-Operation-in-Hypertext-Preprocessor-PHP.pdf.
2. Chen S., Thaduri U. R., Ballamudi V. K. R. Front-end development in react: an overview. Engineering International. 2019. Vol. 7, №2. P. 117-126. URL: https://www.researchgate.net/profile/Venkata-Ballamudi/publication/374154236_Front-End_Development_in_React_An_Overview/links/6510d94f61f18040c220eb13/Front-End-Development-in-React-An-Overview.pdf.
3. Goh H.-A., Ho C.-K., Abas F. S. Front-end deep learning web apps development and deployment: a review. Applied intelligence. 2023. Vol. 53, №12. P. 15923-15945. URL: <https://link.springer.com/article/10.1007/s10489-022-04278-6>.
4. Sharma S., et al. Modern Backend Development Technologies: A Comparative Review and Case Study. International Conference On Emerging Trends In Expert Applications & Security. Singapore: Springer Nature Singapore, 2024. URL: https://link.springer.com/chapter/10.1007/978-981-97-3745-1_12.
5. Suryotrisongko H., Jayanto D. P., Tjahyanto A. Design and development of backend application for public complaint systems using microservice spring boot. Procedia Computer Science. 2017. Vol. 124. P. 736-743. URL: <https://www.sciencedirect.com/science/article/pii/S1877050917329800>.
6. Zulfa I. Pemrograman Web: Hyper Text Markup Language, Cascading Style Sheet, dan Hypertext Preprocessor (PHP). 2024. URL: <https://repository.penerbiteurka.com/media/publications/568736-pemrograman-web-hyper-text-markup-langua-6d8290b4.pdf>.
7. Алтухова А. В., Сагіна Ю. В. До питання про комп'ютерну графіку як основу візуалізації даних. Integration of scientific and modern ideas into practice :

матеріали 8-ї міжнар. наук.-практ. конф. (15-18 листопада 2022 р., Стокгольм, Швеція). 2022. С. 66-72. URL:

<https://books.google.de/books?hl=uk&lr=&id=ijCdEAAAQBAJ&oi=fnd&pg=PA66>.

8. Виганяйло С. М., В'юненко О. Б., В'юненко А. Б. Тенденції розвитку інформаційних технологій у бізнес-аналітиці. Технічні науки та технології. 2021. № 1. С. 103-113. URL:

https://tech.vernadskyjournals.in.ua/journals/2021/1_2021/part_1/10.pdf.

9. Возний О. І. Аналітична система управління фінансами малого підприємства: дис. ... канд. екон. наук. Київ, 2024. URL:

<https://dglb.nubip.edu.ua/server/api/core/bitstreams/93fa62ad-39e0-4492-b770-0645935ebbbb/content>.

10. Грабовський Є. М. Проектування інтелектуального користувацького інтерфейсу систем підтримки електронного навчання : монографія. Харків, 2018. URL:

<https://repository.hneu.edu.ua/jspui/bitstream/123456789/20401/1/%D0%93%D1%80%D0%B0%D0%B1%D0%BE%D0%B2%D1%81%D1%8C%D0%BA%D0%B8%D0%B9%20%D0%84.%D0%9C.pdf>.

11. Гузій В. Т. Розрахункова система для малого бізнесу : кваліфікаційна робота. Київ, 2024. URL: <https://ela.kpi.ua/server/api/core/bitstreams/389541ed-67a4-4329-9198-927553d60aa6/content>.

12. Дамницька А. В. Хмаро орієнтовані платформи, засоби і послуги. New computer technology. 2019. Вип. 17. С. 12-24. URL:

<https://ccjournals.eu/ojs/index.php/nocote/article/view/938>.

13. Данченко О. Б., Бедрій Д. І., Семко О. В. Огляд інформаційних технологій управління бізнес-процесами в організаціях. Управління розвитком складних систем. 2020. № 44. С. 20-26. URL:

<https://er.chdtu.edu.ua/bitstream/ChSTU/2803/1/4.pdf>.

14. Курочка Артур Розробка системи управління даними для для малого бізнесу, на прикладі кав'ярні. Економіка і бізнес: нові реалії, технологічні тренди, цифрові інновації. Economy and business: new realities, technological trends, digital

innovations: збірник матеріалів Міжнародної науково-практичної конференції здобувачів вищої освіти та молодих вчених (Львів-Ірпінь, 15 травня 2025 р.). [Електронне вид.]. Львів: ЛНУ ім. Івана Франка, 2025. 670 с.

15. Кальченко Є. І. Інформаційна система прогнозування ефективності бізнес-проектів: магістерська робота. Суми, 2021. URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/81410/1/master_thesis_Kalchenko.pdf.

16. Карпенко Н. В., Іваннікова М. М., Яловега Н. І., Коваленко Я. А. Інноваційні маркетингові технології для підприємств малого бізнесу. Науковий вісник Полтавського університету економіки і торгівлі. Серія «Економічні науки». 2023. № 3 (109). С. 46-53. URL: <https://www.puet.poltava.ua/index.php/economics/article/view/199>.

17. Катков Ю. І., Звенігородський О. С., Труш О. В., Демидов В. О. Підвищення ефективності малого бізнесу через використання хмарних сервісів. Наукові записки Державного університету інформаційно-комунікаційних технологій. 2020. Вип. 2. С. 34-44. URL: <https://journals.dut.edu.ua/index.php/sciencenotes/article/view/2453>.

18. Куцова Д. Системний аналіз методів розробки інформаційної системи проектування з використанням хмарних обчислень : кваліфікаційна робота. Миколаїв, 2024. URL: <https://eir.nuos.edu.ua/items/b75bf1d6-83de-4500-89a3-bd382fd00deb>.

19. Мінгальова Ю. І. Порівняльний аналіз сучасних систем управління вмістом (CMS) для створення корпоративного сайту. Науковий пошук молодих дослідників : збірник наукових праць студентів, магістрантів та викладачів / за ред. В. М. Єрємєєвої, О. М. Королюк. Житомир, 2012. Вип. 5. С. 168-173.

20. Мітіков М. Ю., Гук Н. А. Дослідження проблем швидкодії програмних додатків. Математичне та комп'ютерне моделювання. Серія: Технічні науки. 2024. С. 22-36. URL: https://www.researchgate.net/profile/Nik-Mitikov/publication/385430727_Investigation_of_Software_Application_Performanc

e_Issues/links/6755f3edb558f41d0fc63121/Investigation-of-Software-Application-Performance-Issues.pdf.

21. Stack Overflow. Рішення щодо реалізації фільтрації форм у HTML і PHP // Stack Overflow. — URL: <https://stackoverflow.com>

22. Лавріщева К. М. Програмна інженерія: підручник. Київ 2008. 315 с. С. 99-115.

23. Олексюк В. П. Основи хмарних технологій : навч. посіб. Тернопіль, 2018

24. Оришака О. В., Марченко А. К. Перспективи використання HTML як інструмента для формування складних текстових документів. Тези доповідей IV Міжнародної науково-практичної конференції «Інформаційна безпека та комп'ютерні технології». Кропивницький, 2021. С. 45-46. URL: <https://kbpz.kntu.kr.ua/file/content/6629/2021-iv-mizhnarodna-naukovo-praktychnoi-konferentsiia-informatsiina-bezpeka-ta-komp-yuterni-tekhnohii-.pdf#page=45>.

25. Пилипюк Т. В., Шліхта Г. О. Використання баз даних для створення сучасних інформаційних систем комерційного призначення. Інформаційні технології в професійній діяльності : матеріали конференції. Рівне, 2019. С. 77-78. URL: <https://iktmvi.rshu.edu.ua/files/konf/ZB-ITVPD-2019.pdf#page=77>.

26. Поперешняк С. В., Вєчерковська А. С. Дослідження розробки вимог до хмарних програм та сервісів. Вісник Херсонського національного технічного університету. 2023. № 4 (87). С. 258-264. URL: <https://cyberleninka.ru/article/n/doslidzhennya-rozrobki-vimog-do-hmarnih-program-ta-servisiv>.

27. Пуцентейло П., Гуменюк О., Довгань Д. Сучасні технології бізнес-аналітики як інструмент для підвищення бізнес-комунікацій компанії. Інститут бухгалтерського обліку, контроль та аналіз в умовах глобалізації. 2022. № 1-2. С. 29-40. URL: <http://ibo.wunu.edu.ua/index.php/ibo/article/viewFile/543/588>.

28. Ткаченко О., Бондар В. Деякі аспекти використання Node.js та MongoDB під час створення освітніх вебзастосунків. Цифрова платформа:

інформаційні технології в соціокультурній сфері. 2023. Т. 6, № 1. С. 76-89. URL: <http://infotech-soccult.knukim.edu.ua/article/view/283958>.

29. Філь Н. Ю., Гавріш А. В. Автоматизація бізнес процесів складського обліку : матеріали конференції. Харків, 2020. URL: <https://api.dspace.khadi.kharkov.ua/server/api/core/bitstreams/a0c0451b-6c26-4bdf-bd0a-c8105a6cafb7/content>.

30. Шендерівська Л. П., Батир А. О. Актуальні завдання стратегічного управління та цифровізації на підприємствах малого бізнесу. Modern science: innovations and prospects : матеріали 14-ї міжнар. наук.-практ. конф. (16-18 жовтня 2022 р., Стокгольм, Швеція). 2022. С. 435-441. URL: <https://sci-conf.com.ua/wp-content/uploads/2022/10/MODERN-SCIENCE-INNOVATIONS-AND-PROSPECTS-16-18.10.22.pdf#page=435>.

31. Ковальчук О. С., Петренко І. В. Сучасні підходи до розробки веб-орієнтованих інформаційних систем для малого бізнесу. Вісник Київського національного університету технологій та дизайну. Серія: Технічні науки. 2023. № 2 (104). С. 89-97. URL: <https://jrnl.knutd.edu.ua/index.php/technical/article/view/1234>.

ДОДАТКИ

Додаток А

Код основних компонентів системи управління даними для кав'ярні

А.1 Файл index.php

```
<?php
// Підключення до бази даних
include 'db.php';

// Запит до бази даних для отримання продуктів
$sql = "SELECT name, image, price FROM products";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // Якщо є продукти, зберігаємо їх у масив
    while($row = $result->fetch_assoc()) {
        $products[] = $row;
    }
} else {
    $products = [];
}

$conn->close();
?>

<!DOCTYPE html>
<html lang="uk">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CRM для кав'ярні</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <!-- Кнопка для відкриття бокового меню -->
  <button class="menu-toggle">&#9776;</button>

  <!-- Бокове меню -->
  <div class="side-menu">
    <ul>
      <li><a href="add_client.php">Додати клієнта</a></li>
      <li><a href="add_product.php">Додати продукт</a></li>
      <li><a href="add_order.php">Додати замовлення</a></li>
    </ul>
  </div>

  <!-- Шапка -->
  <header class="header">
    <div class="logo">
      <h1>CRM для кав'ярні</h1>
    </div>
    <nav class="navbar">
      <ul>
        <li><a href="index.php">Головна</a></li>
```

```

    <li><a href="clients.php">Список клієнтів</a></li>
    <li><a href="products.php">Список продуктів</a></li>
    <li><a href="orders.php">Замовлення</a></li>
</ul>
</nav>
</header>

<!-- Головна секція -->
<section class="hero">
    <div class="hero-text">
        <h2>Ласкаво просимо до нашої кав'ярні!</h2>
        <p>Смачна кава, приємна атмосфера, чудовий сервіс — все, що потрібно
для гарного дня.</p>
    </div>
</section>

<!-- Галерея продуктів -->
<section class="gallery">
    <h2>Наше меню</h2>
    <div class="gallery-grid">
        <?php if (!empty($products)): ?>
            <?php foreach ($products as $product): ?>
                <div class="gallery-item">
                    ">
                    <p><?php echo $product['name']; ?></p>
                    <p>Ціна: <?php echo $product['price']; ?> грн</p>
                </div>
            </?php>
        </?php>
    </div>

```

```

        <?php endforeach; ?>
    <?php else: ?>
        <p>Меню поки що порожнє.</p>
    <?php endif; ?>
</div>
</section>

<!-- Футер -->
<footer>
    <p>&copy; 2025 Кав'ярня CRM. Всі права захищено.</p>
</footer>

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

```

А.2 Файл orders.php

```

<?php
include 'db.php';

// Видалення замовлення
if (isset($_GET['delete'])) {
    $id = intval($_GET['delete']);
    $conn->query("DELETE FROM orders WHERE id = $id");
    header("Location: orders.php");
    exit();
}

```

```

}
?>
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Замовлення</title>
    <link rel="stylesheet" href="orders.css">

</head>
<body>

<h2>Замовлення</h2>

<div class="return-btn">
    <a href="index.php" class="btn">Головна</a>
</div>

<form method="GET">
    <label>Клієнт:</label>
    <select name="client_id">
        <option value="">Bci</option>
        <?php
            $res = $conn->query("SELECT * FROM clients");
            while ($c = $res->fetch_assoc()) {
                $selected = (isset($_GET['client_id']) && $_GET['client_id'] == $c['id']) ?

```

```

"selected" : "";

    echo "<option value='{${c['id']}' $selected>{${c['name']}}</option>";
}
?>
</select>

<label>Продукт:</label>
<select name="product_id">
    <option value="">Бсі</option>
    <?php
        $res = $conn->query("SELECT * FROM products");
        while ($p = $res->fetch_assoc()) {
            $selected = (isset($_GET['product_id']) && $_GET['product_id'] == $p['id'])
? "selected" : "";
            echo "<option value='{${p['id']}' $selected>{${p['name']}}</option>";
        }
    ?>
</select>

<label>Дата від:</label>
<input type="date" name="from_date" value="<?= $_GET['from_date'] ?? " ?>">

<label>до:</label>
<input type="date" name="to_date" value="<?= $_GET['to_date'] ?? " ?>">

<button type="submit" class="btn">Фільтрувати</button>
</form>

```

```

<table>
  <thead>
    <tr>
      <th>Клієнт</th>
      <th>Товар</th>
      <th>К-сть</th>
      <th>Дата</th>
      <th>Дії</th>
    </tr>
  </thead>
  <tbody>
    <?php
      $where = [];

      if (!empty($_GET['client_id'])) {
        $client_id = intval($_GET['client_id']);
        $where[] = "client_id = $client_id";
      }

      if (!empty($_GET['product_id'])) {
        $product_id = intval($_GET['product_id']);
        $where[] = "product_id = $product_id";
      }

      if (!empty($_GET['from_date'])) {
        $from = $_GET['from_date'];
        $where[] = "order_date >= '$from'";
      }
    </?php>
  </tbody>
</table>

```

```

}

if (!empty($_GET['to_date'])) {
    $to = $_GET['to_date'];
    $where[] = "order_date <= '$to'";
}

$where_sql = $where ? "WHERE " . implode(" AND ", $where) : "";

$sql = "SELECT o.*, c.name as client_name, p.name as product_name
        FROM orders o
        JOIN clients c ON o.client_id = c.id
        JOIN products p ON o.product_id = p.id
        $where_sql";

$result = $conn->query($sql);
while ($row = $result->fetch_assoc()) {
    echo "<tr>
        <td data-label='Клієнт'>{$row['client_name']}</td>
        <td data-label='Товар'>{$row['product_name']}</td>
        <td data-label='К-сть'>{$row['quantity']}</td>
        <td data-label='Дата'>{$row['order_date']}</td>
        <td data-label='Дії'>
            <a href='?delete={$row['id']}' onclick=\"return confirm('Видалити це
замовлення?');\" class='delete-btn'>Видалити</a>
        </td>
    </tr>";
}

```

```

    }
    ?>
    </tbody>
</table>

</body>
</html>

```

А.3 Файл products.php

```

<?php
include 'db.php';

$adminPassword = 'admin_crm';
$error = "";

// Обробка запиту на підтвердження видалення
if ($_SERVER['REQUEST_METHOD'] === 'POST' &&
    isset($_POST['confirm_delete'])) {
    $idToDelete = intval($_POST['product_id']);
    $enteredPassword = $_POST['admin_password'];

    if ($enteredPassword === $adminPassword) {
        $deleteQuery = "DELETE FROM products WHERE id = $idToDelete";
        if ($conn->query($deleteQuery)) {
            header("Location: products.php");
            exit();
        }
    }
}

```

```
    } else {  
        $error = "Помилка при видаленні продукту: " . $conn->error;  
    }  
    } else {  
        $error = "Невірний пароль адміністратора."  
    }  
}  
  
// Фільтрація  
$categoryFilter = $_GET['category'] ?? "  
$priceMinFilter = $_GET['price_min'] ?? "  
$priceMaxFilter = $_GET['price_max'] ?? "  
  
$query = "SELECT * FROM products WHERE 1";  
if ($categoryFilter) $query .= " AND category = " . $conn->real_escape_string($categoryFilter) . "";  
if ($priceMinFilter) $query .= " AND price >= " . floatval($priceMinFilter);  
if ($priceMaxFilter) $query .= " AND price <= " . floatval($priceMaxFilter);  
$result = $conn->query($query);  
?>  
  
<!DOCTYPE html>  
<html lang="uk">  
<head>  
    <meta charset="UTF-8">  
    <title>Список продуктів</title>  
    <link rel="stylesheet" href="styles.css">
```

```

    <link rel="stylesheet" href="products.css">
</head>
<body>
<h2>Список продуктів</h2>

<?php if ($error): ?>
    <p style="color:red;"><?php echo $error; ?></p>
<?php endif; ?>

<!-- Форма фільтрації -->
<form method="GET" action="products.php" class="filter-form">
    <label for="category">Категорія:</label>
    <select name="category" id="category">
        <option value="">Бсі</option>
        <?php
            $catRes = $conn->query("SELECT DISTINCT category FROM products");
            while ($cat = $catRes->fetch_assoc()) {
                $selected = ($cat['category'] == $categoryFilter) ? 'selected' : '';
                echo "<option value='{ $cat['category']}'
$selected>{ $cat['category']}</option>";
            }
        ?>
    </select>

    <label for="price_min">Мін. ціна:</label>
    <input type="number" name="price_min" value="<?php echo
htmlspecialchars($priceMinFilter); ?>">

```

```

<label for="price_max">Макс. ціна:</label>

<input type="number" name="price_max" value="<?php echo
htmlspecialchars($priceMaxFilter); ?>">

    <button type="submit" class="btn">Фільтрувати</button>
</form>

<!-- Таблиця продуктів -->
<table>
    <tr>
        <th>Назва</th>
        <th>Категорія</th>
        <th>Ціна</th>
        <th>Фото</th>
        <th>Дії</th>
    </tr>

    <?php while ($row = $result->fetch_assoc()): ?>
        <tr>
            <td><?php echo $row['name']; ?></td>
            <td><?php echo $row['category']; ?></td>
            <td><?php echo $row['price']; ?></td>
            <td></td>
            <td>
                <!-- Форма з підтвердженням пароля -->
                <form method="POST" onsubmit="return confirm('Ви впевнені, що
хочете видалити цей товар?');">
                    <input type="hidden" name="product_id" value="<?php echo $row['id'];

```

```

?>">

        <input type="password" name="admin_password"
placeholder="Пароль" required>

        <button type="submit" name="confirm_delete" class="btn delete-
btn">Видалити</button>

    </form>

</td>

</tr>

<?php endwhile; ?>
</table>

</body>
</html>

```

А.4 Файл db.php

```

<?php
$host = 'localhost';
$user = 'root';
$pass = '';
$db = 'cs_db';

$conn = new mysqli($host, $user, $pass, $db);

if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

?>

```

```
<?php
include 'db.php';

if (isset($_GET['delete'])) {
    $id = intval($_GET['delete']);
    $conn->query("DELETE FROM clients WHERE id = $id");
    header("Location: clients.php");
    exit();
}
?>

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Список клієнтів</title>
    <link rel="stylesheet" href="styles.css">
    <link rel="stylesheet" href="clients.css">
    <style>

    </style>
</head>
<body>
```

```

<h2>Список клієнтів</h2>

<div class="return-btn">
    <a href="index.php" class="btn">Головна</a>
</div>

<table>
    <thead>
        <tr>
            <th>ПІБ</th>
            <th>Email</th>
            <th>Телефон</th>
            <th>Дії</th>
        </tr>
    </thead>
    <tbody>
        <?php
$result = $conn->query("SELECT * FROM clients");
while ($row = $result->fetch_assoc()) {
    echo "<tr>
        <td data-label='Ім'я'>{$row['name']}</td>
        <td data-label='Email'>{$row['email']}</td>
        <td data-label='Телефон'>{$row['phone']}</td>
        <td data-label='Дії'>
            <a href='?delete={$row['id']}' onclick=\"return confirm('Видалити
цього клієнта?');\" class='delete-btn'>Видалити</a>
        </td>
    </tr>
        <?php
    }
}
    </tbody>
</table>

```

```

        </tr>";
    }
    ?>
</tbody>
</table>

</body>
</html>

```

А.6 Файл add_product.php

```

<?php include 'db.php'; ?>
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Додати продукт</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
    <link rel="stylesheet" type="text/css" href="add_product.css">
</head>
<body>

    <h2>Додати новий продукт</h2>
    <!-- Link to go back to the homepage -->
    <ul>

```

```
<li><a href="index.php">Головна</a></li>
</ul>

<form method="POST" action="add_product.php" enctype="multipart/form-
data">

<label for="name">Назва:</label>
<input type="text" name="name" required>

<label for="category">Категорія:</label>
<input type="text" name="category" required>

<label for="price">Ціна:</label>
<input type="number" name="price" required>

<label for="image">Фото:</label>

<input type="file" name="image" required>

<button type="submit" name="add_product">Додати продукт</button>
</form>

<?php
if (isset($_POST['add_product'])) {
    $name = $_POST['name'];
    $category = $_POST['category'];
    $price = $_POST['price'];
    $image = $_FILES['image']['name'];
```

```

$target_dir = "images/";
$target_file = $target_dir . basename($image);

if (move_uploaded_file($_FILES['image']['tmp_name'], $target_file)) {
    $conn->query("INSERT INTO products (name, category, price, image)
VALUES ('$name', '$category', '$price', '$image')");
    echo "<p>Продукт додано успішно!</p>";
} else {
    echo "<p>Помилка при завантаженні зображення.</p>";
}
}
?>

</body>
</html>

```

А.7 Файл add_order.php

```

<?php include 'db.php'; ?>
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Додати замовлення</title>
    <link rel="stylesheet" type="text/css" href="styles.css">

```

```

<link rel="stylesheet" type="text/css" href="add_order.css">
</head>
<body>
  <h2>Додати нове замовлення</h2>
  <!-- Link to the homepage -->
  <ul>
    <li><a href="index.php">Головна</a></li>
  </ul>

  <form method="POST" action="add_order.php">
    <label for="client_id">Клієнт:</label>
    <select name="client_id" required>
      <?php
        $res = $conn->query("SELECT * FROM clients");
        while ($c = $res->fetch_assoc()) {
          echo "<option value='{ $c['id']}'>{ $c['name']}'</option>";
        }
      ?>
    </select>

    <label for="product_id">Продукт:</label>
    <select name="product_id" required>
      <?php
        $res = $conn->query("SELECT * FROM products");
        while ($p = $res->fetch_assoc()) {
          echo "<option value='{ $p['id']}'>{ $p['name']}'</option>";

```

```
    }
    ?>
</select>

<label for="quantity">Кількість:</label>
<input type="number" name="quantity" required>

<label for="order_date">Дата замовлення:</label>
<input type="date" name="order_date" required>

<button type="submit" name="add_order">Додати замовлення</button>
</form>

<?php
if (isset($_POST['add_order'])) {
    $client_id = $_POST['client_id'];
    $product_id = $_POST['product_id'];
    $quantity = $_POST['quantity'];
    $order_date = $_POST['order_date'];

    $conn->query("INSERT INTO orders (client_id, product_id, quantity,
order_date) VALUES ('$client_id', '$product_id', '$quantity', '$order_date')");
    echo "<p>Замовлення додано успішно!</p>";
}
?>
</body>
</html>
```

```
<?php include 'db.php'; ?>
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Додати клієнта</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
  <link rel="stylesheet" type="text/css" href="add_client.css">
</head>
<body>
  <h2>Додати нового клієнта</h2>

  <!-- Link to go back to the homepage -->
  <ul>
    <li><a href="index.php">Головна</a></li>

  </ul>

  <form method="POST" action="add_client.php">
    <label for="name">ПІБ:</label>
    <input type="text" name="name" required>

    <label for="email">Email:</label>
    <input type="email" name="email" required>
```

```

<label for="phone">Телефон:</label>
<input type="text" name="phone" required>

<button type="submit" name="add_client">Додати клієнта</button>
</form>

<?php
if (isset($_POST['add_client'])) {
    $name = $_POST['name'];
    $email = $_POST['email'];
    $phone = $_POST['phone'];

    $conn->query("INSERT INTO clients (name, email, phone) VALUES ('$name',
'$email', '$phone')");

    echo "<p>Клієнта додано успішно!</p>";
}
?>
</body>
</html>

```

А.9 Файл script.js

```

// Відкриття/ закриття бокового меню
const menuToggle = document.querySelector('.menu-toggle');
const sideMenu = document.querySelector('.side-menu');
menuToggle.addEventListener('click', () => {
    sideMenu.classList.toggle('active');
});

```

Стилі CSS для веб-системи управління даними кав'ярні

Б.1 Файл add_client.css

```
body {  
  
    font-family: 'Helvetica', sans-serif;  
  
    background-color: #f9f9f9;  
  
    margin: 0;  
  
    padding: 40px 20px;  
  
    display: flex;  
  
    flex-direction: column;  
  
    align-items: center;  
  
}
```

```
h2 {  
  
    font-size: 26px;  
  
    color: #004e92;  
  
    margin-bottom: 30px;  
  
    text-align: center;  
  
}
```

```
form {
```

```
background-color: #ffffff;

padding: 30px 40px;

border-radius: 16px;

box-shadow: 0 8px 20px rgba(0, 0, 0, 0.1);

width: 100%;

max-width: 400px;

display: flex;

flex-direction: column;

gap: 18px;

}
```

```
label {

    font-size: 14px;

    color: #555555;

}
```

```
input[type="text"],

input[type="email"] {

    padding: 14px;

    font-size: 14px;

    border-radius: 12px;
```

```
border: 1px solid #cccccc;  
  
background-color: #f0f0f0;  
  
outline: none;  
  
transition: border 0.2s ease;  
  
}
```

```
input[type="text"]:focus,  
input[type="email"]:focus {  
  
    border-color: #007bff;  
  
    background-color: #ffffff;  
  
}
```

```
button[type="submit"] {  
  
    padding: 14px;  
  
    font-size: 15px;  
  
    background-color: #007bff;  
  
    color: #ffffff;  
  
    border: none;  
  
    border-radius: 12px;  
  
    cursor: pointer;  
  
    transition: background-color 0.3s ease, transform 0.2s ease;
```

```
}
```

```
button[type="submit"]:hover {  
    background-color: #0056b3;  
    transform: translateY(-1px);  
}
```

```
p {  
    margin-top: 20px;  
    color: green;  
    font-size: 14px;  
    text-align: center;  
}
```

```
/* Style for the "Головна" button */
```

```
ul {  
    text-align: center;  
    margin-top: 20px;  
}
```

```
ul li {  
    list-style-type: none;  
}
```

```
ul li a {  
  
    font-size: 16px;  
  
    color: #007bff;  
  
    text-decoration: none;  
  
    padding: 10px 20px;  
  
    border: 1px solid #007bff;  
  
    border-radius: 6px;  
  
    background-color: #fff;  
  
    transition: background-color 0.3s;  
  
}
```

```
ul li a:hover {  
  
    background-color: #007bff;  
  
    color: #fff;  
  
}
```

Б.2 Файл add_order.css

```
body {  
  
    font-family: 'Helvetica', sans-serif;  
  
    background-color: #f2f2f2;
```

```
margin: 0;

padding: 40px 20px;

display: flex;

flex-direction: column;

align-items: center;

}
```

```
h2 {

    font-size: 26px;

    color: #2c3e50;

    margin-bottom: 30px;

    text-align: center;

}
```

```
form {

    background-color: #ffffff;

    padding: 30px 40px;

    border-radius: 16px;

    box-shadow: 0 8px 20px rgba(0, 0, 0, 0.05);

    width: 100%;

    max-width: 440px;

    display: flex;
```

```
flex-direction: column;  
  
gap: 18px;  
  
}
```

```
label {  
  
    font-size: 14px;  
  
    color: #34495e;  
  
    margin-bottom: 4px;  
  
}
```

```
input[type="number"],  
input[type="date"],  
select {  
  
    padding: 13px;  
  
    font-size: 14px;  
  
    border-radius: 12px;  
  
    border: 1px solid #d0d0d0;  
  
    background-color: #f9f9f9;  
  
    transition: border 0.2s ease;  
  
}
```

```
input:focus,
```

```
select:focus {
```

```
    border-color: #007bff;
```

```
    background-color: #ffffff;
```

```
    outline: none;
```

```
}
```

```
button[type="submit"] {
```

```
    padding: 14px;
```

```
    font-size: 15px;
```

```
    background-color: #007bff;
```

```
    color: #ffffff;
```

```
    border: none;
```

```
    border-radius: 12px;
```

```
    cursor: pointer;
```

```
    transition: background-color 0.3s ease, transform 0.2s ease;
```

```
}
```

```
button[type="submit"]:hover {
```

```
    background-color: #0056b3;
```

```
    transform: translateY(-1px);
```

```
}
```

Продовження додатка Б

```
p {  
    margin-top: 20px;  
    color: green;  
    font-size: 14px;  
    text-align: center;  
}  
  
/* Style the homepage link */  
ul {  
    text-align: center;  
    margin-top: 20px;  
}  
  
ul li {  
    list-style-type: none;  
}  
  
ul li a {  
    font-size: 16px;  
    color: #007bff;  
    text-decoration: none;  
    padding: 10px 20px;
```

```
border: 1px solid #007bff;  
  
border-radius: 6px;  
  
background-color: #fff;  
  
transition: background-color 0.3s;  
  
}
```

```
ul li a:hover {  
  
    background-color: #007bff;  
  
    color: #fff;  
  
}
```

Б.3 Файл add_product.css

```
/* Reset some default styles */  
  
* {  
  
    margin: 0;  
  
    padding: 0;  
  
    box-sizing: border-box;  
  
}
```

```
/* Set the font family for the entire page */
```

```
body {
```

```
    font-family: 'Helvetica', sans-serif;
```

```
    background-color: #f4f4f4;
```

```
    color: #333;
```

```
    padding: 20px;
```

```
}
```

```
/* Style the main container */
```

```
h2 {
```

```
    text-align: center;
```

```
    color: #333;
```

```
    font-size: 28px;
```

```
    margin-bottom: 30px;
```

```
}
```

```
/* Form styles */
```

```
form {
```

```
    background-color: #fff;
```

```
    padding: 40px;
```

```
    border-radius: 8px;
```

```
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
```

Продовження додатка Б

```
max-width: 500px;  
  
margin: 0 auto;  
  
}
```

Продовження додатка Б

```
/* Label styles */  
  
label {  
  
    font-size: 16px;  
  
    margin-bottom: 8px;  
  
    display: block;  
  
    color: #555;  
  
}  
  
  
/* Input field styles */  
  
input[type="text"],  
input[type="number"],  
input[type="file"] {  
  
    width: 100%;  
  
    padding: 12px;  
  
    font-size: 16px;  
  
    border: 1px solid #ccc;  
  
    border-radius: 6px;  
  
    margin-bottom: 20px;
```

```
background-color: #f9f9f9;  
}
```

Продовження додатка Б

```
input[type="text"]:focus,  
input[type="number"]:focus,  
input[type="file"]:focus {  
    outline: none;  
    border-color: #007bff;  
}
```

```
/* Button styles */  
button {  
    width: 100%;  
    padding: 12px;  
    font-size: 16px;  
    background-color: #007bff;  
    color: #fff;  
    border: none;  
    border-radius: 6px;  
    cursor: pointer;  
    transition: background-color 0.3s ease;  
}
```

Продовження додатка Б

```
button:hover {  
    background-color: #0056b3;  
}
```

```
/* Add a success message */
```

```
p {  
    text-align: center;  
    font-size: 18px;  
    margin-top: 20px;  
    color: green;  
}
```

```
/* Error message styles */
```

```
p.error {  
    color: red;  
}
```

```
/* Style for the "Головна" button */
```

```
ul {  
    text-align: center;  
    margin-top: 20px;
```

```
}
```

```
ul li {
```

```
    list-style-type: none;
```

```
}
```

Продовження додатка Б

```
ul li a {
```

```
    font-size: 16px;
```

```
    color: #007bff;
```

```
    text-decoration: none;
```

```
    padding: 10px 20px;
```

```
    border: 1px solid #007bff;
```

```
    border-radius: 6px;
```

```
    background-color: #fff;
```

```
    transition: background-color 0.3s;
```

```
}
```

```
ul li a:hover {
```

```
    background-color: #007bff;
```

```
    color: #fff;
```

```
}
```

Продовження додатка Б

Б.4 Файл clients.css

```
body {  
    font-family: 'Segoe UI', sans-serif;  
    background: #f4f4f9;  
    margin: 0;  
    padding: 20px;  
}
```

```
h2 {  
    text-align: center;  
    color: #333;  
    margin-bottom: 30px;  
}
```

```
.return-btn {  
    text-align: center;  
    margin-bottom: 20px;  
}
```

```
.btn {  
    background-color: #007bff;  
  
    color: white;  
  
    padding: 10px 18px;  
  
    text-decoration: none;  
  
    border-radius: 6px;  
  
    transition: background-color 0.3s ease;  
}
```

```
.btn:hover {  
    background-color: #0056b3;  
}
```

```
table {  
    width: 100%;  
  
    border-collapse: collapse;  
  
    background: white;  
  
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);  
}
```

```
th, td {
```

Продовження додатка Б

```
padding: 14px 16px;  
border-bottom: 1px solid #ddd;
```

Продовження додатка Б

```
text-align: left;  
}
```

```
th {  
    background-color: #007bff;  
    color: white;  
}
```

```
tr:hover {  
    background-color: #f1f1f1;  
}
```

```
.delete-btn {  
    background-color: #dc3545;  
    padding: 6px 12px;  
    color: white;  
    border-radius: 5px;  
    text-decoration: none;  
    font-size: 14px;
```

```
transition: background-color 0.3s ease;  
}
```

```
.delete-btn:hover {
```

```
background-color: #c82333;  
}
```

Продовження додатка Б

```
@media (max-width: 768px) {  
table, thead, tbody, th, td, tr {  
display: block;  
}
```

```
th {  
display: none;  
}
```

```
td {  
position: relative;  
padding-left: 50%;  
border: none;  
border-bottom: 1px solid #ddd;  
}
```

```
td::before {
    content: attr(data-label);
    position: absolute;

    left: 16px;
    font-weight: bold;
    color: #555;
}
}
```

Продовження додатка Б

Б.5 Файл styles.css

```
/* Общие стили */

body {
    font-family: 'Helvetica', sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f9f9f9;
    color: #333;
}

/* Стил для хедера */
```

```
header {  
    background-color: #004e92;  
    color: white;
```

Продовження додатка Б

```
    padding: 20px 0;  
    text-align: center;  
}
```

```
header .logo h1 {  
    margin: 0;  
    font-size: 24px;  
}
```

```
header nav ul {  
    list-style-type: none;  
    padding: 0;  
}
```

```
header nav ul li {  
    display: inline;  
    margin-right: 20px;  
}
```

```
header nav ul li a {  
    text-decoration: none;
```

Продовження додатка Б

```
    font-size: 16px;  
    color: white;  
}
```

```
header nav ul li a:hover {  
    color: #f2f2f2;  
}
```

```
/* Бокове меню */
```

```
.side-menu {  
    position: fixed;  
    top: 0;  
    left: 0;  
    width: 280px;  
    height: 100%;  
    background-color: #004e92;  
    color: white;  
    box-shadow: 4px 0 12px rgba(0, 0, 0, 0.2);
```

```
padding-top: 60px;  
  
transition: transform 0.4s ease-in-out;  
  
transform: translateX(-100%);  
  
}
```

Продовження додатка Б

```
/* Коли меню активовано */
```

```
.side-menu.active {  
  
    transform: translateX(0);  
  
}
```

```
.side-menu ul {  
  
    list-style-type: none;  
  
    padding: 0;  
  
    margin: 0;  
  
}
```

```
.side-menu ul li {  
  
    position: relative;  
  
    margin-bottom: 20px;  
  
}
```

```
.side-menu ul li a {
```

text-decoration: none;

color: white;

font-size: 22px;

Продовження додатка Б

display: block;

padding: 18px 30px;

border-radius: 10px;

text-transform: uppercase;

letter-spacing: 1px;

position: relative;

overflow: hidden;

transition: all 0.3s ease;

}

/* Эффект при наведении на ссылки */

.side-menu ul li a::before {

content: ";

position: absolute;

top: 0;

left: -100%;

width: 100%;

height: 100%;

background-color: #006bb3;

```
transition: all 0.3s ease;  
}
```

Продовження додатка Б

```
.side-menu ul li a:hover::before {  
    left: 0;  
}
```

```
.side-menu ul li a:hover {  
    background-color: #006bb3;  
    transform: translateX(10px);  
}
```

/ Кнопка для відкриття бокового меню */*

```
.menu-toggle {  
    position: absolute;  
    top: 20px;  
    left: 20px;  
    background-color: #004e92;  
    color: white;  
    padding: 18px;  
    font-size: 24px;  
    border: none;
```

```
border-radius: 50%;  
  
cursor: pointer;  
  
transition: all 0.3s ease;
```

Продовження додатка Б

```
z-index: 10;  
  
}
```

```
.menu-toggle:hover {  
  
    background-color: #006bb3;  
  
    transform: rotate(90deg);  
  
}
```

```
/* Стиль для героя */
```

```
.hero {  
  
    background-color: #f1f1f1;  
  
    padding: 40px;  
  
    text-align: center;  
  
}
```

```
.hero-text h2 {  
  
    font-size: 30px;
```

```
color: #333;  
}
```

Продовження додатка Б

```
.hero-text p {  
    font-size: 18px;  
    color: #555;  
}  
  
/* Стиль для галереї */  
.gallery {  
    padding: 20px;  
    text-align: center;  
}  
  
.gallery h2 {  
    font-size: 24px;  
    color: #333;  
}  
  
.gallery-grid {  
    display: flex;  
    justify-content: center;  
    gap: 20px;
```

```
}
```

```
.gallery-grid img {
```

```
    width: 200px;
```

```
    height: 200px;
```

```
    object-fit: cover;
```

```
    border-radius: 10px;
```

```
}
```

```
/* Стил ь для футера */
```

```
footer {
```

```
    text-align: center;
```

```
    padding: 20px;
```

```
    background-color: #004e92;
```

```
    color: white;
```

```
}
```

```
footer p {
```

```
    margin: 0;
```

```
    font-size: 14px;
```

```
}
```

Продовження додатка Б

Б.6 Файл products.css

Продовження додатка Б

```
/* Загальні стилі */  
  
body {  
  
    font-family: 'Helvetica', sans-serif;  
  
    background-color: #f4f4f4;  
  
    margin: 0;  
  
    padding: 20px;  
  
    color: #333;  
  
}  
  
h2 {  
  
    text-align: center;  
  
    color: #222;  
  
    margin-bottom: 20px;  
  
}  
  
/* Кнопка повернення */  
  
.return-btn {  
  
    margin-bottom: 20px;
```

```
text-align: center;  
}
```

Продовження додатка Б

```
.return-btn .btn {  
    background-color: #007bff;  
    color: #fff;  
    text-decoration: none;  
    padding: 10px 18px;  
    border-radius: 8px;  
    transition: background-color 0.3s ease;  
}
```

```
.return-btn .btn:hover {  
    background-color: #0056b3;  
}
```

```
/* Таблиця */
```

```
table {  
    width: 100%;  
    border-collapse: collapse;  
    background-color: #fff;  
    box-shadow: 0 0 12px rgba(0,0,0,0.1);
```

```
overflow-x: auto;  
}
```

Продовження додатка Б

```
th, td {  
    padding: 12px 15px;  
    border-bottom: 1px solid #ddd;  
    text-align: left;  
}
```

```
th {  
    background-color: #007bff;  
    color: #fff;  
}
```

```
tr:hover {  
    background-color: #f1f1f1;  
}
```

```
img {  
    max-width: 100px;  
    height: auto;
```

```
border-radius: 4px;  
}
```

Продовження додатка Б

```
/* Адаптація для мобільних */  
  
@media screen and (max-width: 768px) {  
  
    table, thead, tbody, th, td, tr {  
  
        display: block;  
  
    }  
  
  
  
    thead {  
  
        display: none;  
  
    }  
  
  
  
    tr {  
  
        margin-bottom: 15px;  
  
        background: #fff;  
  
        padding: 10px;  
  
        border-radius: 8px;  
  
        box-shadow: 0 0 5px rgba(0,0,0,0.05);  
  
    }
```

```
td {  
    position: relative;
```

Продовження додатка Б

```
    padding-left: 50%;  
    text-align: left;  
}
```

```
td::before {  
    position: absolute;  
    top: 12px;  
    left: 15px;  
    width: 45%;  
    white-space: nowrap;  
    font-weight: bold;  
    color: #555;  
    content: attr(data-label);  
}
```

```
img {  
    max-width: 100%;  
}
```

```
}  
  
/* Стилi для кнопки видалення */
```

```
.delete-btn {
```

```
background-color: #e74c3c;
```

```
color: white;
```

```
padding: 5px 10px;
```

```
text-decoration: none;
```

```
border-radius: 5px;
```

```
font-size: 14px;
```

```
}
```

```
.delete-btn:hover {
```

```
background-color: #c0392b;
```

```
}
```

```
/* Стилi для форми фiльтрацiї */
```

```
.filter-form {
```

```
margin: 20px 0;
```

```
padding: 10px;
```

```
background-color: #f4f4f4;
```

Продовження додатка Б

```
border-radius: 5px;
```

```
display: flex;
```

```
gap: 10px;
```

Продовження додатка Б

```
flex-wrap: wrap;
```

```
}
```

```
.filter-form label {
```

```
font-size: 14px;
```

```
margin-right: 5px;
```

```
}
```

```
.filter-form input {
```

```
padding: 5px;
```

```
font-size: 14px;
```

```
border-radius: 5px;
```

```
border: 1px solid #ccc;
```

```
}
```

```
.filter-form button {
```

```
background-color: #3498db;
```

```
color: white;
```

```
padding: 8px 15px;  
font-size: 14px;  
border: none;
```

Продовження додатка Б

```
border-radius: 5px;  
cursor: pointer;  
}
```

```
.filter-form button:hover {  
    background-color: #2980b9;  
}
```

Б.7 Файл orders.css

```
body {  
    font-family: Arial, sans-serif;  
    background: #f4f4f4;  
    padding: 20px;  
}  
  
h2 {
```

```
text-align: center;  
}
```

```
.return-btn {
```

Продовження додатка Б

```
margin-bottom: 20px;  
text-align: center;  
}
```

```
.btn, .delete-btn {  
background-color: #007bff;  
color: white;  
padding: 6px 12px;  
text-decoration: none;  
border-radius: 4px;  
transition: background 0.3s;  
margin: 2px;  
}
```

```
.btn:hover {  
background-color: #0056b3;  
}
```

```
.delete-btn {  
background-color: #dc3545;  
}
```

```
.delete-btn:hover {  
    background-color: #c82333;  
}
```

Продовження додатка Б

```
form {  
    margin-bottom: 20px;  
    display: flex;  
    flex-wrap: wrap;  
    gap: 10px;  
    justify-content: center;  
    align-items: center;  
}  
  
form label {  
    margin-right: 5px;  
}  
  
table {  
    width: 100%;  
    border-collapse: collapse;  
    background: white;  
}  
  
table, th, td {  
    border: 1px solid #ddd;
```

```
}
```

```
th, td {
```

```
    padding: 10px;
```

```
    text-align: center;
```

```
}
```

```
@media (max-width: 768px) {
```

```
    table, thead, tbody, th, td, tr {
```

```
        display: block;
```

```
    }
```

```
    td {
```

```
        position: relative;
```

```
        padding-left: 50%;
```

```
        text-align: left;
```

```
    }
```

```
    td::before {
```

```
        content: attr(data-label);
```

```
        position: absolute;
```

```
        left: 10px;
```

```
        font-weight: bold;
```

```
    }
```

```
}
```

Продовження додатка Б

Сертифікат з міжнародної науково-практичної конференції студентів та молодих учених "Інформаційні технології в економіці та управлінні"



