МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Крентя Ігоря Сергійовича
(прізвище, ім'я, по батькові здобувача)

на тему **Розробка програмного забезпечення замовлення та видачі ліків**
(повна назва теми)

за матеріалами праць провідних спеціалістів з розробки ПЗ та проектування БД
(повна назва бази дослідження)

науковий керівник к. е. н. доцент _____ Баран С. В.
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 11.06.2025р. № 12
Завідувач кафедри _____
(підпис)

д.т.н., професор Зеленський О.С.
Наук. ступень, вчене звання *Ініціали, прізвище*

Кривий Ріг – 2025

ЗГОДА здобувача вищої освіти

Державного університету економіки і технологій про перевірку кваліфікаційної роботи на прояви академічного плагіату та розміщення в Репозитарії Університету

Я, Крентъ Ігор Сергійович, підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота Розробка програмного забезпечення замовлення та видачі ліків виконана самостійно та не містить академічного плагіату. Я не надавав і не одержував недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлений. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформований, що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлений.

Дата підпис ініціали, прізвище (власноруч)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ/факультет	Інформаційних технологій
Кафедра	Інформатики і прикладного програмного забезпечення
Спеціальність	Інженерія програмного забезпечення
Форма навчання	Денна

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____ Зеленський О.С.
(підпис) (Прізвище, ініціали)
« 11 » червня 2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

1. Тема роботи «Розробка програмного забезпечення замовлення та видачі ліків»

Керівник роботи к. е. н. доцент Баран С. В.
затверджені наказом закладу вищої освіти від «04» квітня 2025 р. № 222-ст

2. Строк подання здобувачем роботи до «09» червня 2025 р.

3. Зміст кваліфікаційної роботи, об'єкт, предмет та мета дослідження:

Розділ 1. Постановка задачі

Розділ 2. Проектування системи

Розділ 3. Проектування бази данних Web-сайту

Розділ 4. Розробка системи

Об'єкт дослідження: аптека

Предмет дослідження: алгоритми замовлення та видачі ліків

Мета кваліфікаційної роботи: *розробка програмного забезпечення замовлення та видачі ліків*

5. Дата видачі завдання «04» квітня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МДР	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	04.04.2025-13.04.2025	
2	Підготовка розділу 2	14.04.2025-26.04.2025	
3.	Підготовка розділу 3	27.04.2025-15.05.2025	
4	Підготовка розділу 4	16.05.2025-08.06.2025	
5	Реєстрація завершеної кваліфікаційної роботи	09.06.2025	Реєстраційний №___ «09»червня 2025 р.
6	Отримання відгуку від наукового керівника	10.06.2025	
7	Подання кваліфікаційної роботи на перегляд завідувачу кафедри	11.06.2025	
8	Отримання зовнішньої рецензії	12.06.2025	
9	Попередній захист кваліфікаційної роботи на кафедрі	13.06.2025	
10	Підготовка до захисту в ЕК	16.06.2025-21.06.2025	

Завдання підготував науковий керівник _____ ___Баран С. В.___
(підпис) (прізвище та ініціали)

Завдання одержав _____ ___Крентъ І. С.
_
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

на кваліфікаційну бакалаврську роботу

«Розробка програмного забезпечення замовлення та видачі ліків»

Крентя Ігоря Сергійовича

Кваліфікаційна бакалаврська робота на здобуття освітньо-кваліфікаційного рівня бакалавра зі спеціальності 121 «Інженерія програмного забезпечення» – Державний університет економіки і технологій – Кривий Ріг, 2025.

У бакалаврській дипломній роботі розроблене програмне забезпечення замовлення та видачі ліків: замовлення ліків через сайт аптеки, зміна статусу замовлення через програмний додаток. Програмний додаток розроблено на мові C++ з використанням бібліотеки OpenGL на основі MFC для роботи з тривимірною графікою та технології Node.js для роботи з базами даних на сайті.

Ключові слова: АПТЕКА, ЗАМОВЛЕННЯ, ЛІКИ, СТАТУС, САЙТ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД(база даних)	Впорядкований набір логічно взаємопов’язаних даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів.
СУБД	Система управління базами даних.
ПЗ	Програмне забезпечення.
MFC(Microsoft Foundation Classes)	Дає можливість розробляти GUI-застосунки для Microsoft Windows на мові C++.

ЗМІСТ

ВСТУП

У сучасному світі цифрові технології дедалі глибше проникають у різні сфери суспільного життя, медицина та фармацевтика не виключення. В умовах стрімкого розвитку інформаційних технологій та постійного зростання попиту на якісне медичне обслуговування виникає потреба у впровадженні інноваційних рішень, здатних підвищити ефективність взаємодії між пацієнтами, медичними працівниками та фармацевтами. Одним із перспективних напрямів є автоматизація процесу замовлення та видачі лікарських засобів за допомогою спеціалізованих інтерактивних програмних систем. Впровадження такої системи у сферу фармакології покращує якість обслуговування аптеки, зменшує затримки у видачі ліків.

Існуючі традиційні методи обслуговування в аптеках часто є неефективними під час великого потоку клієнтів. Вони супроводжуються чергами, неточностями у прийомі замовлень, не обходиться без складності в швидкому пошуку необхідних медикаментів. У багатьох випадках відсутність зручних інструментів для візуального контролю наявності товару та стану замовлення призводить до дезінформації клієнтів і зниження рівня довіри до аптечного сервісу. Саме тому розробка інтерактивної системи, яка поєднує автоматизовану обробку замовлень з елементами графічної візуалізації, є актуальним.

Інтерактивні інтерфейси та візуальні компоненти програмного забезпечення надають зручності користування системою та роблять її зрозумілою. Вони дають змогу швидко орієнтуватися в інформації, швидко приймати рішення, також знижують навантаження на користувальницьку частину системи, ке моде виникнути через необхідність обробки великої кількості даних. Такий підхід сприяє полегшенню та пришвидшенню праці персоналу аптеки, зменшенню кількості помилок у замовленнях та покращенню взаємодії з клієнтами.

Метою цієї дипломної роботи є розробка інтерактивної системи замовлення та видачі ліків з використанням графічної візуалізації, яка повинна

модернізувати процес обслуговування клієнтів в аптечній інфраструктурі, забезпечити структуру замовлень та спростити взаємодію між користувачем і системою. Передбачається, що така система зможе об’єднати функції зберігання, обробки й візуального представлення інформації у єдиному інтерфейсі, доступному як для працівників аптек, так і для клієнтів.

Об’єктом дослідження є процес замовлення, обробки та видачі лікарських засобів у рамках діяльності аптечних закладів. У фокусі уваги перебувають як внутрішні бізнес-процеси, пов’язані з рухом медикаментів, так і зовнішня взаємодія із кінцевим користувачем.

Предметом дослідження виступають методи побудови інтерактивних програмних інтерфейсів, моделі обробки замовлень та алгоритми графічної візуалізації інформації у контексті аптечного обслуговування. Досліджуватиметься також рактичний аспект реалізації таких систем, їх зручність, функціональність та потенціал до розширення у майбутньому.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ

1.1. Характеристика задачі

Завдання створення інтерактивної системи замовлення та видачі лікарських засобів виникає як відповідь на комплекс практичних проблем, що спостерігаються в сучасних аптечних мережах. Основні труднощі пов’язані з обмеженою швидкістю обслуговування, неефективною комунікацією між клієнтом і фармацевтом, браком доступу до інформації про наявність препаратів, а також високим ризиком помилок при обробці замовлень у пікові години. За умов зростання попиту на аптечні послуги, особливо в періоди сезонних епідемій або кризових ситуацій, ці проблеми стають критичними, адже без достатнього цифрового інструментарію аптека втрачає можливість обслуговувати велику кількість клієнтів без втрат у якості.

Сутність задачі полягає у створенні програмного рішення, яке дозволяє об’єднати у єдиній системі процеси пошуку, замовлення, обробки, контролю і видачі медикаментів. Для цього необхідно передбачити декілька логічно пов’язаних модулів, які виконуватимуть функції взаємодії з клієнтом, управління даними про препарати, обслуговування запитів та формування звітності. Відповідно до вимог задачі, система має забезпечувати можливість здійснення запиту на медикаменти із заданими параметрами, швидкої перевірки наявності препарату, відображення результатів пошуку з детальною інформацією, додавання товарів до замовлення, відстеження статусів обробки і підтвердження отримання товару після його підготовки працівником аптеки.

Функціональна складність задачі також полягає в необхідності синхронізації даних у реальному часі між клієнтським і службовим інтерфейсом. Якщо клієнт подає запит на отримання конкретного лікарського засобу, система повинна одразу перевірити, чи не було його зарезервовано іншими користувачами, або перевірити чи не змінився залишок у базі даних з моменту останнього оновлення. Це вимагає реалізації структурованої логіки

транзакцій і контролю доступу до необхідних даних, щоб уникнути конфліктів і дублювання замовлень.

Важливе місце в системі посідає графічна складова. Відображення інформації у вигляді зрозумілих візуальних блоків дає змогу зменшити час на обробку інформації й зробити інтерфейс більш доступним для користувачів різного віку та з різним досвідом користування цифровими сервісами. Користувач має бачити чітку картину про стан замовлення без потреби читати довгі описи чи чекати підтвердження працівника. У службовій частині інтерфейсу передбачаються інструменти для візуального контролю потоку замовлень, автоматичне оновлення статусів, можливість відсортувати запити за пріоритетністю або часом подачі.

З технічної точки зору, задача охоплює розробку структури бази даних, реалізацію API для взаємодії між фронтендом і бекендом, побудову логіки роботи з користувачами, розмежування рівнів доступу, організацію процедури авторизації та захисту персональних даних. Кожна транзакція в системі повинна залишати запис у журналі подій, що дозволяє здійснювати аудит та виявляти помилки на будь-якому етапі замовлення або видачі. При цьому інтерфейс не повинен створювати надмірного навантаження на серверну частину — важливо забезпечити баланс між функціональністю та швидкодією.

Окремо слід відзначити необхідність реалізації механізмів фільтрації та сортування медикаментів за різними критеріями: назвою, діючою речовиною, виробником, формою випуску, терміном придатності. Це дозволить користувачу знайти потрібний препарат й швидко оцінити альтернативи, обрати доступніші або відповідніші варіанти. Система повинна підтримувати введення часткових запитів, орфографічну нечутливість, автозаповнення та підказки.

Складовою задачі є і реалізація управління товарними залишками, при якому відбувається фіксація кожного списання, резервування, надходження нової партії, а також облік повернень. Дані про залишки повинні оновлюватися миттєво після кожної дії — як зі сторони замовлення клієнта, так і моменту

видачі ліків працівником. У разі збоїв або затримок у синхронізації виникає ризик некоректного обслуговування або втрати довіри до системи.

Серед додаткових вимог до системи є вимоги до підтримки багатомовного інтерфейсу, відповідність стандартам доступності, можливість роботи на пристроях із різною роздільною здатністю, мінімальні залежності від сторонніх сервісів. Продукт повинен бути готовим до розгирення функціоналу та можливостей в майбутньому, повинен бути передбачений доступ до інтеграції з мобільними додатками або електронними медичними системами.

1.2. Огляд існуючих web-сайтів

Під час аналізу характеристики задачі було оглянуто існуючі веб-сайти аптек та способи реалізації механізмів замовлення, які можуть надати ці сайти. Найбільшу увагу привернув портал Копійка, оскільки процес замовлення ліків на сайті являє собою інтегровану систему, що дозволяє користувачам швидко і зручно оформлювати замовлення лікарських засобів через інтернет. Система надає користувачам можливість вибору ліків з широкого асортименту, доступного на порталі, а також дозволяє оформити доставку чи самовивіз з аптек, що співпрацюють з платформою.

Процес замовлення починається з реєстрації користувача на порталі або входу до вже існуючого облікового запису (Рис. 1.1.).

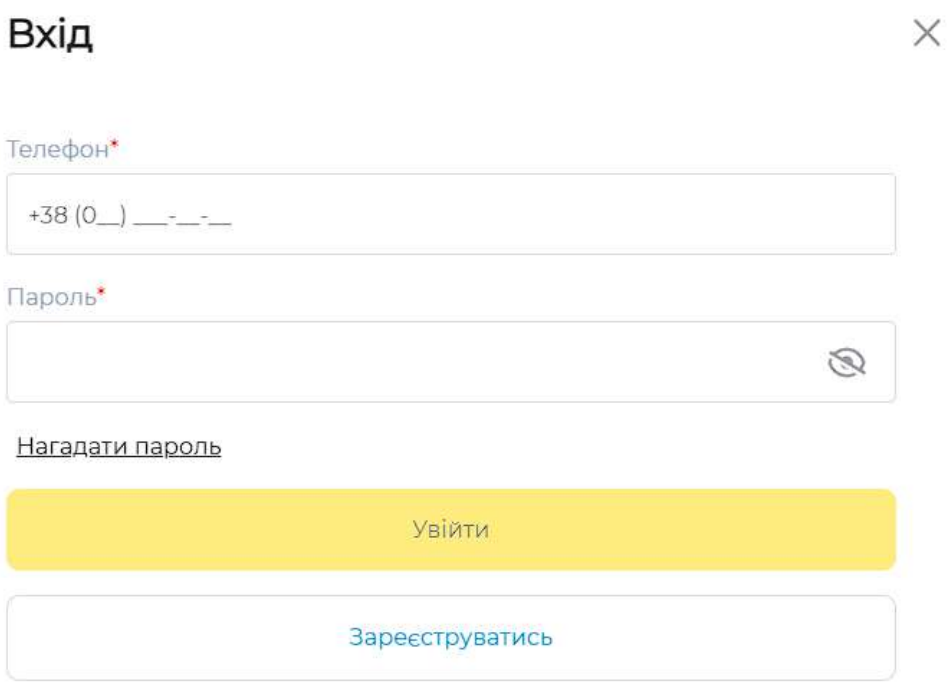

Рис. 1.1. Вхід на портал Копійка

Після цього клієнт має доступ до електронного каталогу ліків, який містить детальну інформацію про кожен товар, включаючи ціну, наявність на складі, дозування та можливі заміни (Рис. 1.2.).

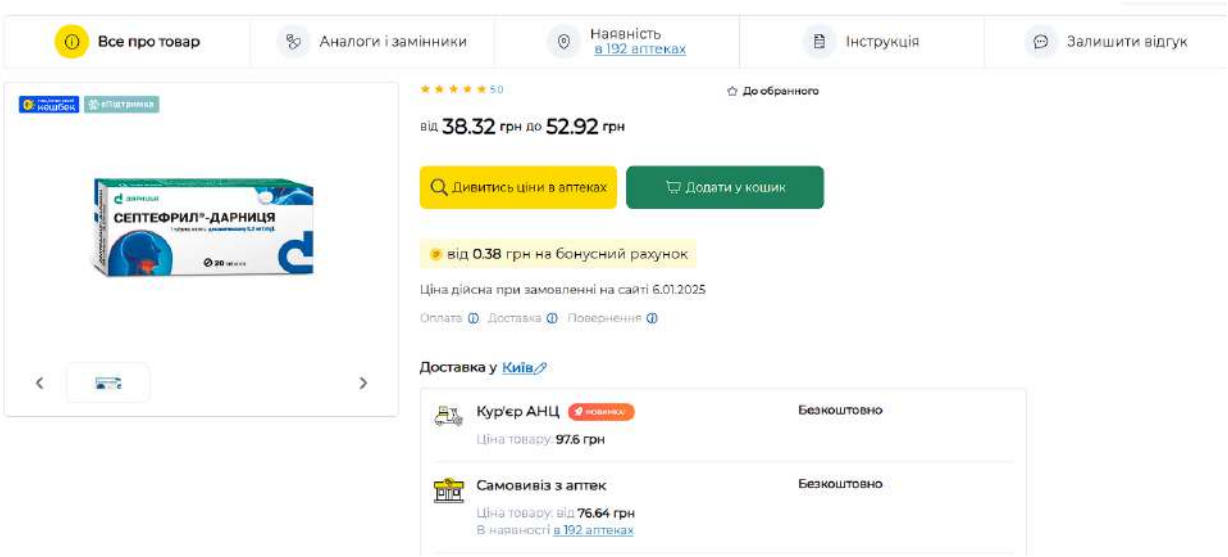

Рис. 1.2. Інформація про товар

На веб-порталі «Копійка» реалізовано функціональну та зручну систему фільтрації товарів. Сайт пропонує для вибору велику кількість товарів, тому система фільтрації необхідна, щоб можна було зручно орієнтуватися в асортименті. Користувач може скористатися такими фільтрами, як:

- категорія товару,
- форма випуску,
- діюча речовина,
- торгова назва,
- виробник,
- країна походження,
- ціновий діапазон,
- наявність у конкретних аптеках,
- акційні пропозиції або знижки,
- рейтинг або відгуки користувачів.

Це спрощує пошук і дає змогу зосередитися саме на тих товарах, які відповідають індивідуальним потребам.

Після вибору ліків, клієнт додає їх до кошика, де можна змінювати кількість одиниць кожного препарату або видаляти непотрібні товари. На цьому етапі система автоматично розраховує загальну суму замовлення, а також додає можливі додаткові витрати на доставку або інші послуги (Рис. 1.3).

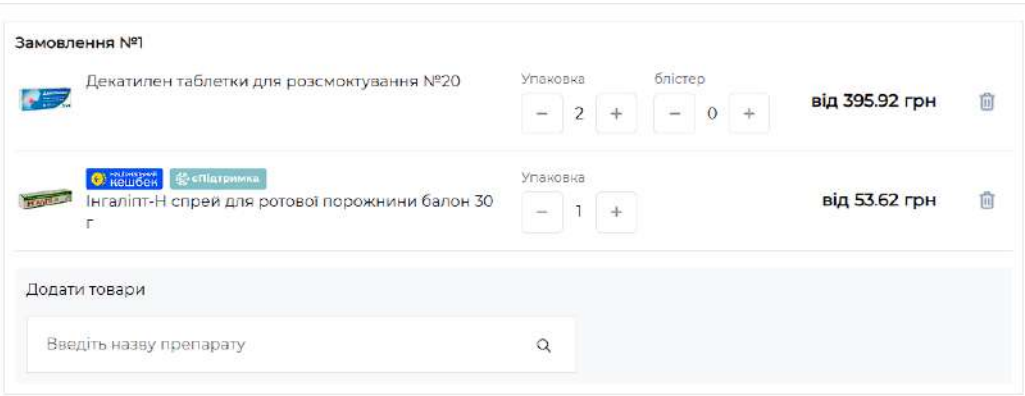

Рис. 1.3. Кошик з товаром

Далі клієнт переходить до оформлення замовлення, де вводить необхідні контактні дані, адреси для доставки, а також вибирає спосіб оплати (Рис. 1.4.). Портал Копійка підтримує різні методи оплати, зокрема, онлайн-банкінг, картки, оплату при отриманні.

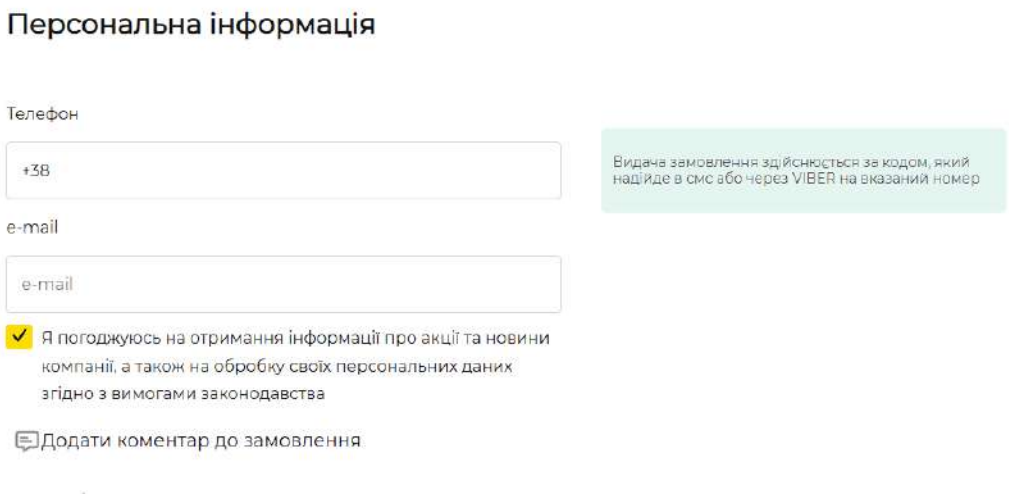

Рис. 1.4. Замовлення товару

Замовлення ліків на порталі Копійка також включає функціонал для управління відстеженням статусу замовлення, що дозволяє клієнту бути в курсі етапів виконання замовлення – від обробки до доставки або готовності до самовивозу. Портал гарантує актуальність даних про наявність товарів і надає користувачеві всі необхідні інструменти для зручної та безпечної покупки ліків онлайн.

Після підтвердження замовлення, система генерує відповідну документацію та інформує користувача про статус замовлення, який можна відслідковувати в особистому кабінеті, а також починається процес створення замовлення у базі даних.

Аналіз веб-порталу «Копійка» показує добре продуману інтеграцію інтерфейсних рішень, які орієнтовані на різну аудиторію користувачів. Процес замовлення ліків побудований таким чином, що він не виманає окремих інструкцій або пояснення, користувач проходить зрозумілий шлях, у якому кожен крок технологічно підкріплений відповідною функціональністю. Усі необхідні кнопки для здійснення замовлення на виду, і на них легко звернути увагу.

Сайт вирізняється не лише широким асортиментом, а й дійсно зручною навігацією завдяки фільтрам, що охоплюють як фармацевтичні характеристики товарів, так і бізнес-параметри: доступність у конкретних

точках, наявність знижок, рейтинги виробників тощо. Це робить сервіс адаптивним до різних моделей мотивів користувачів — від тих, хто шукає конкретний препарат, до тих, хто порівнює варіанти за ціновими або брендовими критеріями.

Кошик, фінальний розрахунок, вибір способу доставки та оплати — все реалізовано з урахуванням стандартів електронної комерції, з додатковими елементами контролю та відстеження замовлень. Система сама надає користувачу актуальну інформацію про наявність препаратів, а також дозволяє змінювати параметри замовлення ще до його підтвердження.

Також варто відзначити технологічну стабільність процесу: усі дані передаються в базу з логічною структурою, що дозволяє надалі можливість керувати замовленнями у іншій програмі, зберігати історію покупок і гарантувати точність обробки.

Реалізований функціонал сайту «Копійка» може слугувати взірцем при проектуванні програмного забезпечення для замовлення та видачі ліків в частині візуальної організації інформації, багатошарової фільтрації та інтерактивного супроводу користувача на кожному етапі роботи з сервісом.

1.3. Аналіз вимог до веб-сайту

Предметною областю даного дослідження є процес організації замовлення та видачі лікарських засобів у фармацевтичних установах. Сучасні аптеки виконують функції не лише точок продажу медикаментів, а й інформаційних центрів, де клієнт очікує отримання достовірної, актуальної та зрозумілої інформації щодо препаратів, умов їх використання та наявності. При цьому зростає потреба у впровадженні технологічних рішень, які забезпечують підтримку повного циклу взаємодії між клієнтом і аптекою в цифровому середовищі.

Основні учасники предметної області наступні:

- Клієнт (покупець лікарських засобів) — особа, яка формує запит на придбання медикаменту, шукає його за назвою або призначенням, перевіряє наявність, подає замовлення та отримує препарат;
- Фармацевт (працівник аптеки) — особа, що відповідає за обробку замовлень, консультування клієнтів, контроль залишків, комплектування замовлень та їх видачу;
- Система управління аптекою — внутрішня цифрова інфраструктура, яка містить дані про препарати, замовлення, залишки, користувачів і логіку їх взаємодії.

Процес взаємодії в межах предметної області умовно поділяється на декілька ключових етапів:

- Формування запиту на препарат: клієнт вводить пошукову інформацію — назву, активну речовину або симптом — після чого система пропонує варіанти з наявного асортименту.
- Вибір і уточнення інформації: клієнт знайомиться з описом препарату, ціною, формою випуску, умовами зберігання. Важливою є наявність супровідної інформації, яка допомагає зробити усвідомлений вибір.
- Оформлення замовлення: після вибору препаратів клієнт формує замовлення, вказує бажаний спосіб отримання та ідентифікує себе (авторизація).
- Опрацювання замовлення працівником аптеки: фармацевт перевіряє наявність препаратів, резервує їх, комплектує замовлення та змінює його статус у системі.
- Видача замовлення: клієнт отримує повідомлення про готовність і звертається до аптеки для отримання свого замовлення. Факт видачі фіксується у системі.

Окремої уваги заслуговує проблема структури й оновлення інформації про залишки препаратів. У багатьох аптеках ці дані ведуться вручну або із затримкою оновлення, що призводить до ситуацій, коли клієнт замовляє препарат, якого вже немає в наявності. Автоматизовані системи дозволяють

вирішити це завдяки безперервному контролю складських операцій та миттєвому оновленню даних.

Ще одним аспектом є різноманіття медикаментів за торговими назвами, формами, дозуваннями, упаковками. Це вимагає впровадження у систему розгалуженої класифікації, яка дозволяє користувачу швидко знайти потрібний варіант та не помилитися при виборі. Для цього потрібно впровадити системи категоризації за АТС-кодами, наявністю рецепту, типом діючої речовини тощо.

У традиційних аптеках частина цієї роботи покладається на працівника, який мусить швидко орієнтуватися в асортименті, пам’ятати, що є на складі, відповідати на запитання клієнтів, паралельно обслуговуючи декілька людей. Це створює значне когнітивне навантаження, особливо у години пікового попиту. Водночас клієнт у таких умовах не завжди має змогу повноцінно оцінити альтернативи чи самостійно прийняти рішення — він змушений покладатися на усну консультацію, яка часто обмежена в часі або деталізації.

Тому інтерактивна система повинна забезпечити функції, які зазвичай виконуються людиною, але в електронному вигляді. Каталог із товарами, контекстна інформація, підказки, відображення статусів, можливість самостійного уточнення наявності без потреби ставити запитання працівнику.

Слід також враховувати нормативні обмеження. Замовлення та продаж певних категорій медикаментів дозволяється лише за наявності рецепта. У зв’язку з цим у межах предметної області постає завдання організації перевірки документів або верифікації прав користувача на замовлення таких препаратів. Це може включати ручну перевірку, прикріплення рецепта в цифровому форматі або інтеграцію з електронною системою охорони здоров’я.

Серед додаткових об'єктів предметної області — класифікатори ліків, складські одиниці, партії постачання, терміни придатності, персональні дані клієнтів, електронна історія замовлень, статуси замовлень, повідомлення про

зміни, механізми зворотного зв’язку. Усі ці об’єкти формують логічну модель, на основі якої будується функціональна архітектура системи.

Висновки до розділу 1

У першому розділі було розглянуто загальні аспекти та логіку розробки інтерактивної системи замовлення та видачі ліків з графічною візуалізацією. Основною метою системи є автоматизація процесу замовлення і видачі ліків, покращення взаємодії між аптеками та їх клієнтами, зниження ймовірність помилок видачі та полегшення навантаження на фармацевтів.

Було виокремлено два основні компоненти системи: вебінтерфейс для клієнтів, десктоп-додаток для аптекарів, який передбачає систему візуалізації ліків у 3D. Кожен з компонентів виконує специфічні функції, при цьому їх взаємодія гарантує якісну обробку замовлень і швидку видачу препаратів.

Основна увага була приділена характеристиці задачі, яка полягає в інтеграції різних рівнів логіки та інформаційних потоків системи, а також забезпеченні високої точності та зручності для кінцевого користувача. Вибір технологій та інструментів для розробки був обґрунтований необхідністю створення гнучкої, адаптивної та зручної в обслуговуванні системи, що забезпечує якісну роботу як для користувачів, так і для аптекарів.

Таким чином, в результаті аналізу предметної області, задач і характеристик системи були визначені ключові вимоги до її реалізації, що закладають основу для подальшого етапу розробки.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Архітектура системи

Архітектура розробленої системи стало наступмним етапом проектування, оскільки визначає загальний каркас системи, її компоненти та взаємодії між ними. Для реалізації інтерактивної системи замовлення та видачі ліків з графічною візуалізацією використовується багаторівнева архітектура, яка включає три основних компоненти: клієнтський веб-інтерфейс, десктоп-додаток для аптекарів і базу даних. Кожен з цих компонентів виконує свою роль, взаємодіючи з іншими через чітко визначену серверну логіку, яка забезпечує стабільну роботу системи.

Клієнтська частина системи реалізована у вигляді вебінтерфейсу, який дозволяє користувачам взаємодіяти з системою через браузер. Основні функції вебінтерфейсу включають пошук ліків, оформлення замовлень, перевірку їх статусів та сповіщення про готовність замовлення. Вебінтерфейс є точкою взаємодії між кінцевим користувачем і системою.

Для забезпечення динамічного інтерфейсу використовуються мова програмування JavaScript та фреймворк React. Ці технології дозволяють створити інтерактивний інтерфейс, що реагує на дії користувача в реальному часі, без необхідності перезавантаження сторінки. Наприклад, при пошуку ліків система може миттєво відображати результати пошуку, а також оновлювати дані про наявність чи ціну препарату, без перезавантаження всієї сторінки.

Система взаємодіє з серверною частиною через RESTful API, яке відповідає за передачу даних між фронтендом і бекендом. Це дозволяє веб-інтерфейсу отримувати актуальну інформацію про наявність ліків, статуси замовлень та інші дані, забезпечуючи актуальність відображених даних для кінцевого користувача.

Десктоп-додаток призначений для аптекарів і виконує роль обробки замовлень, підтвердження їх виконання та візуалізації фізичного розташування ліків на полиці. Він дозволяє аптекарю побачити, які замовлення потрібно виконати, а також переглядати аптечну полицю у вигляді інтерактивної 3D-візуалізації.

Десктоп-додаток реалізований на C++, оскільки ця мова програмування забезпечує високу продуктивність, це враховується через необхідність для роботи з 3D-графікою в реальному часі [4]. Для реалізації 3D-візуалізації використовується OpenGL, інструмент для рендерингу 3D-графіки і дає змогу ефективно відображати складні візуальні об'єкти, такі як упаковки ліків на полиці.

Комунікація між десктоп-додатком і сервером здійснюється через RESTful API, що дозволяє отримувати актуальні дані про замовлення та стан ліків. Кожне замовлення отримує унікальний ідентифікатор, що дозволяє точно прив'язати його до конкретного препарату та забезпечити правильну видачу.

Серверна частина системи є головним компонентом, який відповідає за обробку запитів від клієнтського інтерфейсу та десктоп-додатку. Вона містить серверну логіку для обробки даних, їх зберігання в базі даних та взаємодію з іншими компонентами системи.

Для реалізації серверної частини використовується Node.js — технологія, яка забезпечує високу продуктивність і здатна обробляти великі обсяги запитів від користувачів [1]. На сервері працює логіка, що забезпечує прийом та обробку запитів на оформлення замовлень, перевірку наявності ліків, оновлення статусів замовлень і передачу відповідних даних клієнтському інтерфейсу або десктоп-додатку.

Для забезпечення комунікації між компонентами через API використовуються стандарти REST, що дозволяють отримувати необхідні дані в структурованому вигляді.

Для структуризації інформації була побудована схема архітектури системи (Рис. 2.1.). Архітектура системи передбачає чітку взаємодію між усіма компонентами через API:

- Клієнтський інтерфейс звертається до серверної частини для отримання даних про ліки, формування замовлення та перевірки статусу.
- Серверна частина обробляє запити та передає актуальні дані з бази даних, підтримуючи зв'язок з обома компонентами.
- Десктоп-додаток аптеки отримує інформацію про замовлення через API, оновлюючи дані про статуси замовлень та виконуючи візуалізацію полиць.

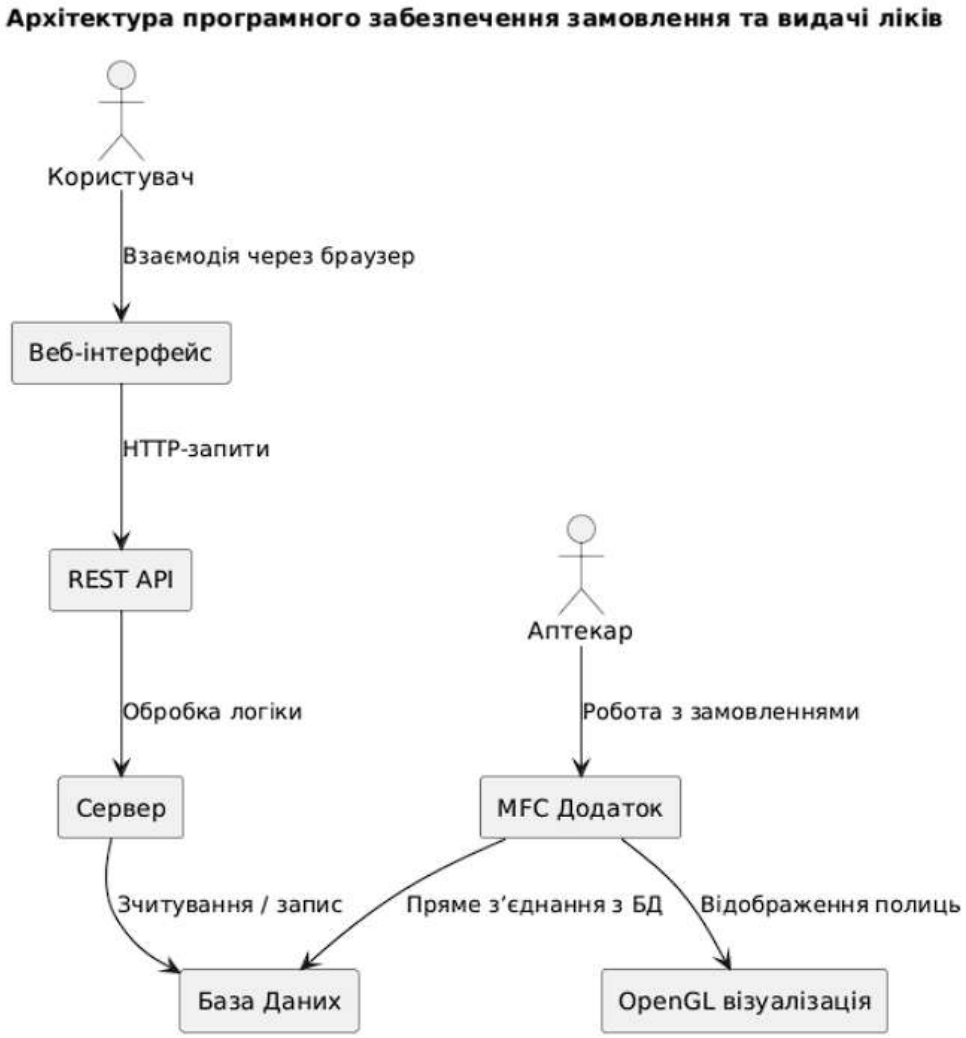

Рис. 2.1. Схема архітектури системи

Завдяки такій архітектурі, система забезпечує зручний доступ до всіх необхідних функцій для кінцевих користувачів, аптекарів та адміністраторів.

2.2. Алгоритм роботи системи

Алгоритм роботи системи замовлення та видачі ліків з графічною візуалізацією описує послідовність операцій, які виконуються при взаємодії користувачів із системою, починаючи від пошуку ліків до видачі замовлення аптекарем. Всі компоненти системи взаємодіють між собою через сервер, що обробляє запити та здійснює необхідні операції з базою даних. Оскільки система має кілька учасників — кінцевих користувачів, аптекарів і серверну частину — алгоритм роботи системи включає кілька основних етапів, кожен з яких є частиною процесу обробки замовлення.

Користувач заходить на вебсайт аптеки та здійснює пошук ліків за допомогою введення ключових слів у поле пошуку, вибору категорії або фільтрів. Сервер обробляє запит і повертає список ліків, що відповідають критеріям пошуку, включаючи відомості про кожен препарат: назву, форму випуску, ціну та наявність. Після того, як користувач знайшов потрібні ліки, він додає їх до кошика. Вебінтерфейс передає інформацію про обрані ліки та кількість на сервер. Система перевіряє для кожного препарату його наявність на складі. Користувач переходить до оформлення замовлення, вказуючи адресу доставки, контактні дані та інші реквізити. Якщо всі необхідні ліки є в наявності, замовлення оформлюється та зберігається в базі даних зі статусом «Очікує на обробку».

Після створення замовлення воно стає доступним для аптекаря через десктоп-додаток. Аптекар бачить перелік нових замовлень, вибирає необхідне, переглядає його склад та місце розташування препаратів на віртуальній полиці. Після збору замовлення він підтверджує видачу, і статус змінюється на «Виконано». Вся інформація оновлюється в базі даних, а користувач отримує відповідне сповіщення.

З цього можна визначити основні дії які виконують користувачі системи, які стануть основою для алгоритму системи.

1. Дії користувача наступні (Рис. 2.2.):
    a. Вхід на вебсайт
    b. Введення ключових слів у пошуку
    c. Вибір категорії або фільтра
    d. Надсилання пошукового запиту
    e. Отримання результатів із бази даних
    f. Ознайомлення з інформацією про ліки
    g. Додавання препаратів до кошика
    h. Вказання кількості кожного препарату
    i. Перевірка наявності на складі
    j. Перехід до оформлення замовлення
    k. Заповнення контактної інформації та адреси
    l. Підтвердження замовлення
2. Дії серверної частини мають бути такі (Рис. 2.3.):
    a. Прийом запитів із вебінтерфейсу
    b. Обробка пошукових запитів
    c. Отримання даних з бази
    d. Формування відповіді користувачу
    e. Перевірка наявності ліків
    f. Створення нового замовлення
    g. Присвоєння статусу
    h. Збереження замовлення у базі даних

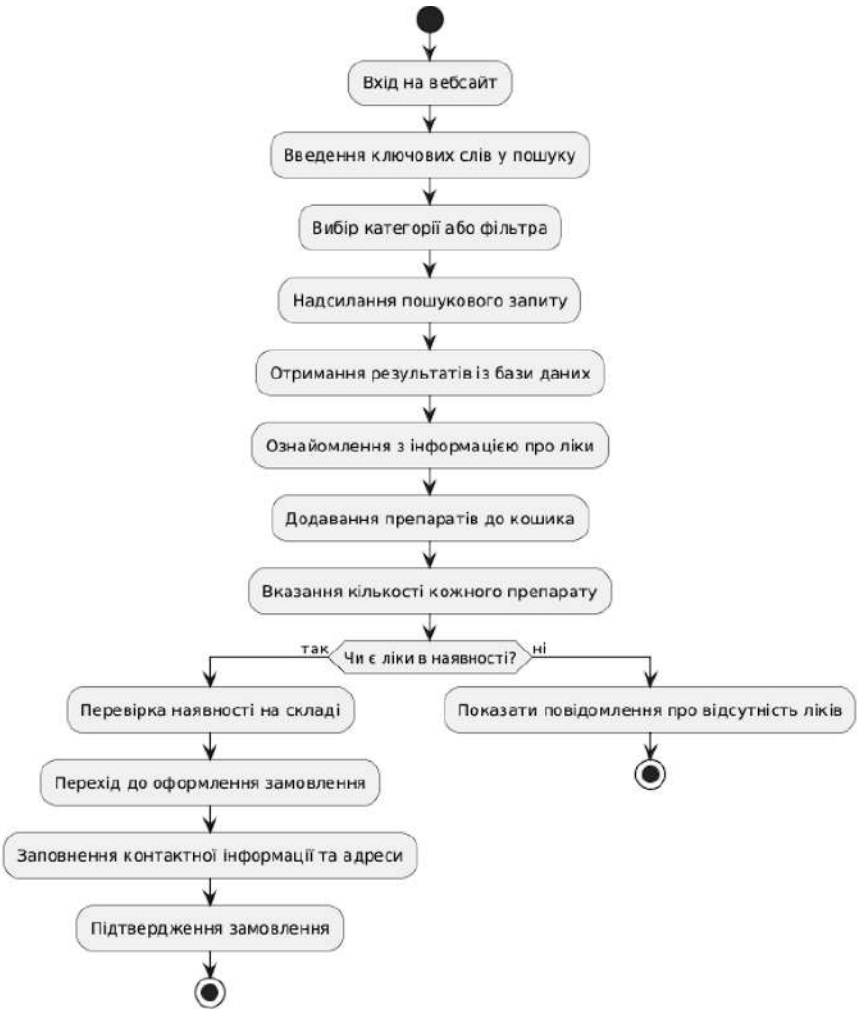

Рис. 2.2. Дії користувача на сайті

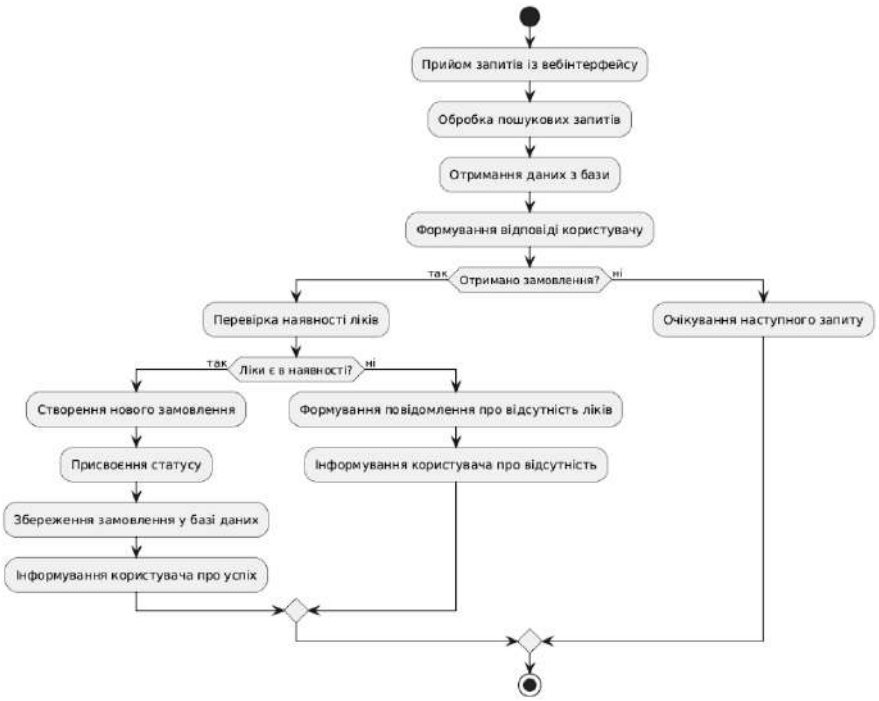

Рис. 2.3. Дії серверної частини

3. Дії десктоп-додатку для аптекаря (Рис. 2.4.):

a. Отримання нового замовлення з бази даних
b. Відображення складу замовлення
c. Відкриття 3D-полиці
d. Пошук потрібних препаратів
e. Підтвердження збору
f. Оновлення статусу замовлення
g. Внесення змін до бази даних
h. Надсилання повідомлення клієнту

Рис. 2.4. Дії десктоп-додатку

4. Взаємодія з базою даних:

a. Зберігання відомостей про ліки
b. Зберігання користувацьких даних
c. Збереження замовлень та їх статусів
d. Надання даних серверу та додатку
e. Синхронізація змін між компонентами

Висновки до розділу 2

У розділі було розглянуто архітектуру розробленої системи, що базується на багаторівневому підході і включає клієнтський веб-інтерфейс, десктоп-додаток для аптекарів та базу даних. Визначено ключові компоненти та їх взаємодію через серверну логіку і RESTful API. Аналіз алгоритмів показав чітку послідовність дій для кожного з компонентів, що гарантує коректне виконання основних функцій — від пошуку і замовлення ліків кінцевим користувачем до обробки замовлень і візуалізації їх стану для аптекарів. Такий структурований підхід сприяє підвищенню зручності користування, оперативності обробки інформації та зменшенню ризиків помилок, що є важливим для ефективної роботи аптечного сервісу. Загалом, обрана архітектура і розроблені алгоритми відповідають поставленим вимогам і створюють надійну основу для подальшого розвитку і впровадження системи.

РОЗДІЛ 3
ПРОЕКТУВАННЯ БАЗИ ДАНИХ WEB-САЙТУ

3.1. Створення моделі бази даних

Після характеристики задачі та побудови алгоритму системи була створена інфологічна модель бази даних.

Інфологічна модель бази даних у рамках даного дипломного проекту забезпечує структуроване представлення предметної області системи замовлення та видачі ліків. Вона фокусується на ключових сутностях, їхніх атрибутах та взаємозв’язках, що дає змогу формалізувати інформацію незалежно від конкретної реалізації системи управління базами даних.

Атрибути сутностей визначають їх властивості та мають різні параметри [18]. Формат атрибутів відповідає типам даних: текстові значення представлені рядками фіксованої або змінної довжини, числові — цілими або дробовими числами, а булеві — двійковими значеннями. Обов’язковість заповнення атрибутів регламентується їх роллю в системі — ідентифікатори та ключові поля є необхідними, тоді як додаткові поля можуть бути факультативними для врахування розширеної інформації. Доступ до атрибутів розмежовується залежно від ролі користувача: конфіденційні або системні дані доступні лише адміністраторам, а оператори працюють із стандартними полями. Частота використання атрибутів визначається функціональними потребами — ключові атрибути, такі як унікальні ідентифікатори, використовуються найчастіше, тоді як допоміжні поля застосовуються рідше. Відображення значень атрибутів у звітах та інтерфейсах регулюється відповідними налаштуваннями, що забезпечує зручність користування. Допустиме дублювання значень стосується лише атрибутів, які не мають унікальності за своєю природою, наприклад, контактних номерів або адрес. Обмеження на введення значень передбачають формати, діапазони та спеціалізовані списки допустимих даних. Атрибути виконують різні функції — від ідентифікації та опису до підтримки аналітичних процесів.

Сутності, що моделюють об’єкти предметної області, мають визначені способи звернення до їхніх екземплярів, рівень структурної активності, обмеження доступу та потенційну кількість записів. Унікальні ідентифікатори забезпечують однозначну ідентифікацію кожного екземпляра. Динамічні сутності, такі як замовлення або клієнти, активно змінюються та доповнюються в процесі роботи системи, у той час як статичні сутності, наприклад, статуси або типи ліків, оновлюються рідше. Рівень доступу до сутностей регулюється ролями користувачів, а кількість екземплярів визначається фізичними ресурсами системи та специфікою предметної області.

Зв’язки між сутностями відображають логічні залежності та потік інформації в системі. Назви зв’язків уточнюють їх семантику, наприклад, асоціація між замовленням і клієнтом. Напрямок зв’язку може бути одностороннім або двостороннім залежно від логіки предметної області. Упорядкування екземплярів підпорядкованих сутностей здійснюється відповідно до контексту — за хронологією, пріоритетом чи іншими критеріями. Класи членства визначають, які екземпляри беруть участь у зв’язках з урахуванням предметно-логічних обмежень.

Об’єкт **Клієнт** має ключові атрибути, які забезпечують унікальність і цілісність даних. Атрибут **Код_клієнта** має формат цілого числа, є обов’язковим і слугує первинним ключем. Його область допустимих значень — додатні цілі числа, а дублювання значень виключене. Цей атрибут використовується постійно, зокрема в зв’язках з іншими сутностями, і виводиться у звітах як основний ідентифікатор клієнта.

Атрибут **ПІБ** має формат текстового поля з обмеженням до 20 символів. Він обов’язковий, його значення часто використовується в роботі бази даних і відображається в інтерфейсі для користувачів. Область допустимих значень обмежена іменами клієнтів, дублювання можливе, оскільки різні клієнти можуть мати однакові імена. Роль атрибута — надання описової інформації для ідентифікації клієнта.

Атрибут **Телефон** має текстовий формат із максимальним обсягом у 13 символів, що дозволяє зберігати міжнародні телефонні номери. Він є обов’язковим, використовується часто, виводиться у звітах і може дублюватися між клієнтами, якщо вони надають однаковий номер. Область значень обмежена валідними телефонними номерами. Його роль полягає у забезпеченні контактної інформації клієнта.

Атрибут **Електронна_пошта** має текстовий формат з обмеженням до 50 символів і є необов’язковим. Він використовується для збереження електронної адреси клієнта, яка може бути корисною для комунікації або розсилки повідомлень. Область допустимих значень включає валідні електронні адреси. Дублювання можливе, якщо кілька клієнтів користуються однією поштовою скринькою. Роль атрибута — забезпечення можливості комунікації з клієнтом електронною поштою.

Атрибут **Дата_реєстрації** має формат дати, є обов’язковим. Він зберігає дату створення облікового запису клієнта в системі. Область допустимих значень охоплює валідні дати, дублювання допустиме для клієнтів, які зареєстровані в один день. Роль атрибута полягає у забезпеченні інформації про час початку співпраці з клієнтом і може бути використаною для аналітики чи сегментації клієнтської бази.

Об’єкт **Відділення** містить дані про філіали, які обслуговують клієнтів. Атрибут **Код_відділення** має формат цілого числа і є обов’язковим первинним ключем. Його область значень охоплює додатні числа, дублювання виключене. Цей атрибут використовується часто, особливо у зв’язках з іншими сутностями, та є основним ідентифікатором.

Атрибут **Назва** має текстовий формат із довжиною до 50 символів. Він обов’язковий, часто використовується для відображення інформації про відділення, дублювання можливе. Область значень включає назви відділень, а роль атрибута — забезпечення інформації для розрізнення філіалів.

Атрибут **Адреса** також представлений у текстовому форматі з обмеженням до 50 символів. Він обов’язковий, часто використовується для

відображення місця розташування відділення. Дублювання можливе у разі спільного розташування кількох служб. Його область значень охоплює реальні поштові адреси.

Атрибут **Телефон** необов’язковий, має текстовий формат із обмеженням до 13 символів. Він використовується для надання контактної інформації відділення, може дублюватися у разі однакових номерів для кількох філіалів.

Атрибут **Графік_роботи** має текстовий формат до 50 символів, є необов’язковим. Він відображає робочі години відділення, значення можуть дублюватися для різних закладів.

Об’єкт **Товар** зберігає інформацію про продукти. Атрибут **Код_товару** представлений у форматі цілого числа, є обов’язковим первинним ключем. Область значень обмежується додатними числами, дублювання виключено. Атрибут часто використовується у зв’язках, виводиться для ідентифікації продуктів.

Атрибут **Назва** має формат текстового поля до 50 символів, обов’язковий і часто використовується для відображення описової інформації. Значення можуть дублюватися у випадку схожих товарів.

Атрибут **Категорія** представлений у текстовому форматі до 20 символів, є необов’язковим. Він відображає класифікацію товарів за типом, значення можуть повторюватися.

Атрибут **Ціна** має числовий формат із двома десятковими знаками, є обов’язковим і використовується для розрахунків. Область значень обмежується невід’ємними числами.

Атрибут **Кількість_на_складі** є цілим числом, обов’язковий, часто використовується у звітах і транзакціях. Значення не повинні бути негативними.

Атрибут **Рецепт_потрібен** має булевий формат і визначає, чи потрібен рецепт для товару. Він є обов’язковим і має два допустимих значення: "Так" або "Ні".

Об’єкт **Замовлення** відображає транзакції між клієнтами та відділеннями. Атрибут **Код_замовлення** є цілим числом, обов’язковим і виконує роль первинного ключа. Його значення унікальне, не допускає дублювання і є основою для зв’язків з іншими сутностями. Частота використання висока, область допустимих значень обмежена додатними числами.

Атрибут **Код_клієнта** є зовнішнім ключем, що посилається на Клієнт. Формат — ціле число, обов’язковий. Він використовується для зв’язування замовлення з конкретним клієнтом. Дублювання можливе, якщо кілька замовлень належать одному клієнту.

Атрибут **Код_відділення** також є зовнішнім ключем і вказує на Відділення. Його формат — ціле число, обов’язковий, область значень відповідає наявним кодам відділень. Атрибут часто використовується для зв’язків і звітів.

Атрибут **Дата_замовлення** має формат дати, обов’язковий для відображення часу оформлення. Область значень охоплює всі валідні дати, дублювання допустиме. Його роль — відстеження часу створення замовлення.

Атрибут **Код_статусу** є зовнішнім ключем, що вказує на сутність Статус. Формат — ціле число, обов’язковий. Його область значень визначається доступними статусами, дублювання можливе у разі однакових статусів для різних замовлень.

Атрибут **Сума** представлений у числовому форматі з двома десятковими знаками, обов’язковий і використовується для фінансових розрахунків. Область допустимих значень включає невід’ємні числа, дублювання можливе для замовлень із однаковою вартістю.

Об’єкт **Деталі_замовлення** визначає конкретні товари, пов’язані з певними замовленнями. Атрибут **Код_деталей** є цілим числом, обов’язковим первинним ключем. Він має унікальні значення, що не дублюються, і використовується для ідентифікації записів.

Атрибут **Код_замовлення** є зовнішнім ключем, що посилається на сутність Замовлення. Його формат — ціле число, обов’язковий, дублювання допустиме, якщо кілька товарів належать одному замовленню.

Атрибут **Код_товару** також є зовнішнім ключем, що вказує на сутність Товар. Його формат — ціле число, обов’язковий. Область допустимих значень відповідає наявним товарам у базі, дублювання можливе у разі однакових товарів у різних деталях замовлень.

Атрибут **Кількість** має формат цілого числа, є обов’язковим. Його значення не повинно бути від’ємним, а дублювання можливе для однакових товарів у різних замовленнях. Роль атрибута полягає в забезпеченні кількісного відображення товарів.

Атрибут **Ціна** є числовим із двома десятковими знаками, обов’язковий. Він відповідає за фінансові розрахунки і дублюється, якщо вартість товарів у деталях однакова. Область допустимих значень включає невід’ємні числа.

Об’єкт **Оплата** відображає фінансові транзакції, пов’язані з замовленнями. Атрибут **Код_оплати** є цілим числом, обов’язковим і виконує роль первинного ключа. Його значення унікальні, дублювання виключене.

Атрибут **Код_замовлення** є зовнішнім ключем, що посилається на Замовлення. Формат — ціле число, обов’язковий, частота використання висока. Область допустимих значень відповідає кодам замовлень, дублювання не допустиме для кількох оплат одного замовлення.

Атрибут **Дата_оплати** має формат дати, є обов’язковим. Він відображає момент виконання оплати, область допустимих значень охоплює валідні дати, дублювання можливе.

Атрибут **Сума** представлений у числовому форматі з двома десятковими знаками, є обов’язковим. Його значення може дублюватися, оскільки різні замовлення можуть мати однакову суму, і обмежуються невід’ємними числами.

Атрибут **Спосіб_оплати** має текстовий формат до 50 символів, є обов’язковим. Він вказує тип фінансової операції, дублювання допустиме для однакових способів оплати.

Об’єкт **Статус** описує поточний стан замовлення і використовується для його класифікації. Атрибут **Код_статусу** є цілим числом, обов’язковим і виконує роль первинного ключа. Його значення унікальні, дублювання виключене. Роль атрибута полягає у забезпеченні унікальної ідентифікації статусу для подальшого використання в інших об’єктах, таких як Замовлення.

Атрибут **Назва** представлений у текстовому форматі з обмеженням до 20 символів, є обов’язковим. Він відображає назву статусу, наприклад, "Новий" або "Обробляється". Область допустимих значень включає текстові дані, дублювання можливе, якщо однаковий статус застосовується до кількох замовлень. Роль атрибута полягає у забезпеченні описової інформації для спрощення управління замовленнями.

Зв’язок між сутностями **Клієнт і Замовлення** називається "Оформляє". Він є одностороннім, рух іде від клієнта до замовлення, упорядкування екземплярів виконується за часом створення замовлень.

Зв’язок між **Замовленням і Відділенням** має назву "Виконується". Рух односторонній, упорядкування екземплярів підпорядкованого об’єкта здійснюється за відділеннями.

Зв’язок між **Замовленням і Деталями_замовлення** має назву "Містить". Він є одностороннім, екземпляри впорядковуються за товарами, пов’язаними з конкретним замовленням.

Зв’язок між **Замовленням і Оплатою** має назву "Оплачується". Рух односторонній, екземпляри впорядковуються за датою виконання оплат.

Зв’язок між сутностями **Відділення і Замовлення** називається "Виконується". Він є одностороннім, рух здійснюється від відділення до замовлення. Упорядкування екземплярів виконується за часом оформлення замовлень, які обслуговує конкретне відділення.

Зв’язок між **Товаром і Деталями_замовлення** має назву "Має". Він є одностороннім, рух здійснюється від товару до деталей замовлення. Упорядкування екземплярів відбувається за товарами, що входять до конкретних деталей замовлень.

Зв’язок між **Статусом і Замовленням** має назву "Відображає". Він є одностороннім, рух здійснюється від статусу до замовлення. Упорядкування екземплярів виконується за поточним станом кожного замовлення, відповідно до призначеного статусу.

Після розробки та аналізу інфологічної моделі була створена логічна модель для обліку інтернет-замовлень ліків (Рис. 3.1.).

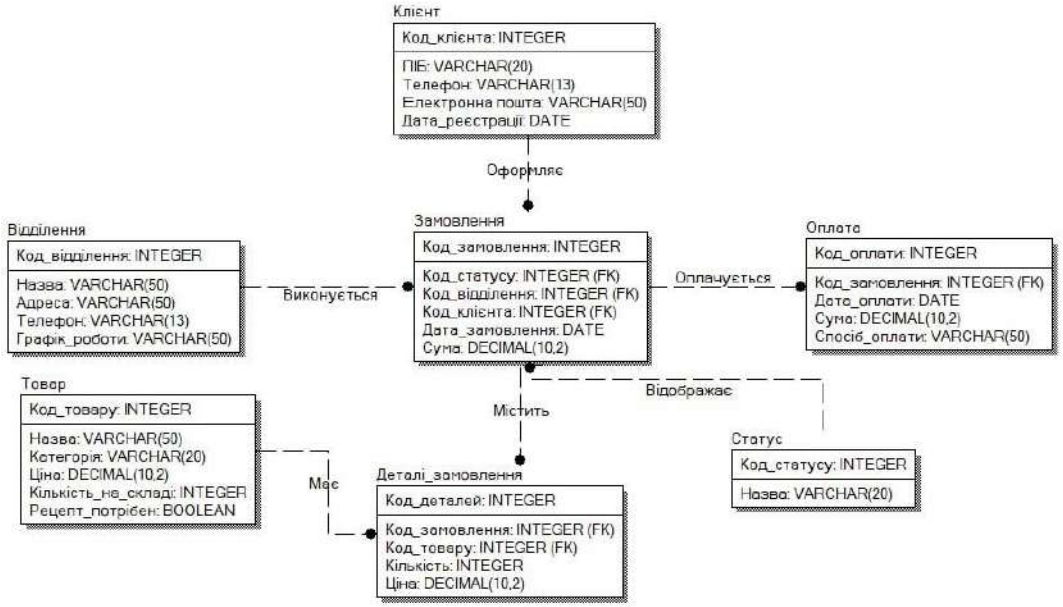

Рис. 3.1. Логічна модель для обліку інтернет-замовлень ліків

Логічна модель даних для бази даних видачі замовлень ліків описує організацію сутностей, їх атрибутів, зв’язків між ними та обмежень. У моделі виділено сутності, що відповідають основним елементам системи, таким як клієнти, відділення, товари, замовлення, деталі замовлень, оплати та статуси.

Сутність **Клієнт** описує користувачів системи, які здійснюють замовлення. Вона має атрибути, що включають унікальний ідентифікатор клієнта, який виконує роль первинного ключа, повне ім’я, контактний телефон, електронну пошту, яка може бути відсутньою, та дату реєстрації. Ця

сутність забезпечує можливість відстеження історії замовлень кожного клієнта.

Сутність **Відділення** містить інформацію про пункти обслуговування. Ключовим атрибутом є унікальний ідентифікатор відділення. До інших атрибутів належать назва, адреса, контактний телефон, а також графік роботи. Ця сутність використовується для управління логістикою та визначення місця обробки замовлень.

Сутність **Товар** описує об’єкти, доступні для замовлення. Її атрибути включають унікальний ідентифікатор товару, назву, категорію, ціну, кількість на складі та ознаку того, чи потрібен рецепт. Ця сутність дозволяє вести облік доступного асортименту та здійснювати перевірку наявності товарів перед оформленням замовлення.

Сутність **Замовлення** представляє замовлення товару в інтернеті. Вона має атрибути, що включають унікальний ідентифікатор замовлення, ідентифікатор клієнта, ідентифікатор відділення, дату створення замовлення, ідентифікатор статусу замовлення та загальну суму. Вона забезпечує централізоване управління всіма замовленнями та є основою для зв’язків із деталями замовлень та оплатами.

Сутність **Деталі_замовлення** деталізує склад кожного замовлення, описуючи товари та їх кількість. Вона включає атрибути, такі як унікальний ідентифікатор деталей замовлення, ідентифікатор замовлення, ідентифікатор товару, кількість та ціну. Ця сутність дозволяє забезпечити точне визначення того, які товари входять у кожне замовлення.

Сутність **Оплата** зберігає дані про фінансові транзакції, пов’язані із замовленнями. Її атрибути включають унікальний ідентифікатор оплати, ідентифікатор замовлення, дату оплати, суму та спосіб оплати. Вона використовується для відстеження надходження коштів і обробки оплат.

Сутність **Статус** визначає поточний стан замовлення. Вона містить унікальний ідентифікатор статусу та його назву, яка відображає етап

виконання замовлення, наприклад, «Прийнято» або «Виконано». Ця сутність дозволяє вести облік і змінювати статуси замовлень.

Зв’язки між сутностями в логічній моделі визначаються асоціаціями типу «один до багатьох» та «багато до багатьох»: сутність **Клієнт** пов’язана з кількома екземплярами сутності **Замовлення**, проте кожен екземпляр сутності Замовлення асоційований лише з одним клієнтом (зв’язок Оформляє), сутність **Замовлення** має зв’язок із сутністю **Деталі_замовлення**, що дозволяє одному замовленню включати кілька деталей, але кожна деталь належить до одного замовлення (зв’язок Містить). Зв’язок між сутностями **Замовлення** і **Відділення** визначається як «один до багатьох», де одне замовлення виконується лише одним відділенням, але кожне відділення може обробляти багато замовлень (зв’язок Виконується).

Зв’язок між сутністю **Замовлення і Оплата** також є «один до багатьох»: одне замовлення може бути оплачено кількома способами або частинами, але кожна оплата стосується лише одного замовлення (зв’язок Оплачується).

Сутність **Замовлення** пов’язана із сутністю **Статус** зв’язком «один до одного», де кожне замовлення може мати лише один поточний статус, і кожен статус прив’язаний до одного замовлення (зв’язок Відображає).

Зв’язок між сутностями **Товар і Деталі_замовлення** визначається як «один до багатьох»: кожен товар може бути присутнім у багатьох деталях замовлення, але кожна деталь замовлення містить лише один конкретний товар (зв’язок Має).

У моделі також враховано обмеження на забезпечення унікальності первинних ключів для кожної сутності та дотримання цілісності зовнішніх ключів, які забезпечують коректне посилання між пов’язаними сутностями.

Після розробки та аналізу логічної моделі була створена фізична модель для обліку інтернет-замовлень ліків (Рис. 3.2.).

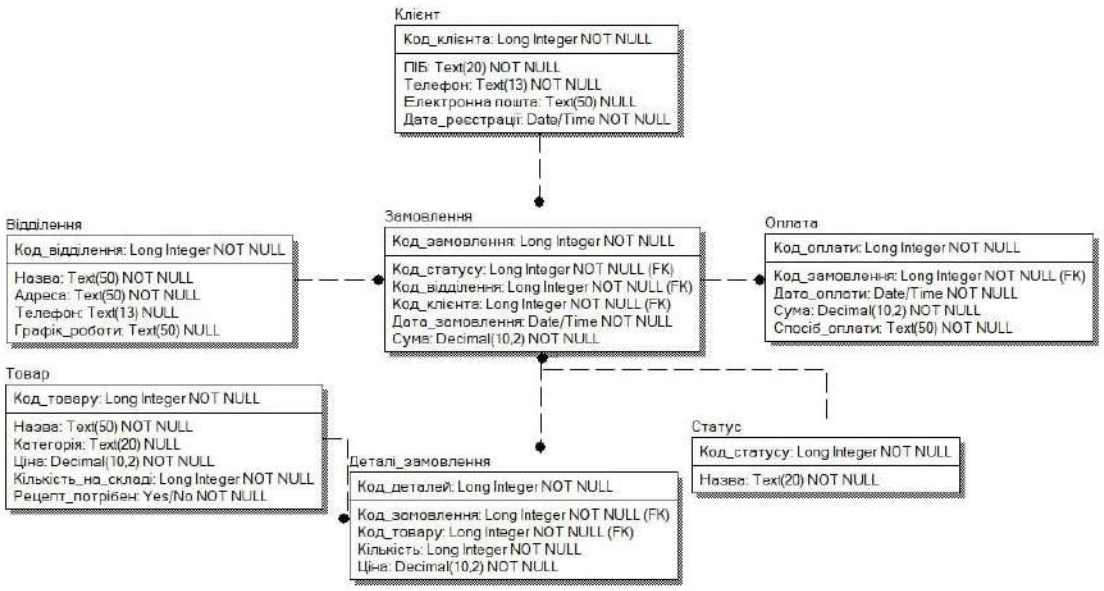

Рис. 3.2. Фізична модель для обліку інтернет-замовлень ліків

Фізична модель бази даних відображає логічну структуру на рівні її реалізації в СУБД, враховуючи оптимізацію продуктивності, забезпечення цілісності даних і зручність доступу. Таблиці виступають як матеріалізовані сутності, де кожна таблиця має набір стовпчиків, що відповідають атрибутам сутностей. Ідентифікатори (первинні ключі) реалізуються за допомогою типу даних INT із використанням механізму автоматичного прирощення (AUTO_INCREMENT) для дотримання унікальності.

Формати даних вибрано відповідно до вимог точності та ефективності використання пам’яті. Текстові атрибути використовують тип NVARCHAR з урахуванням кодування Unicode, числові значення зберігаються як DECIMAL для фінансових даних або INT для цілочисельних величин. Поля, що зберігають дати, представлені типом DATE, що забезпечує коректність і стандартизацію форматів.

Цілісність даних реалізована через первинні ключі (PRIMARY KEY), які гарантують унікальність записів у таблицях. Зовнішні ключі (FOREIGN KEY) встановлюють зв’язки між таблицями та забезпечують дотримання референційної цілісності. Обов’язкові поля, такі як дати або суми, мають

обмеження NOT NULL, що виключає можливість зберігання порожніх значень. Поля, які можуть бути відсутніми, допускають значення NULL.

3.2. Створення бази даних

Враховуючи особливості та специфіку СУБД MySql та поставлені задачі, була створена структура бази даних для бази даних для системи замовлення та видачі ліків (Рис. 3.3.)

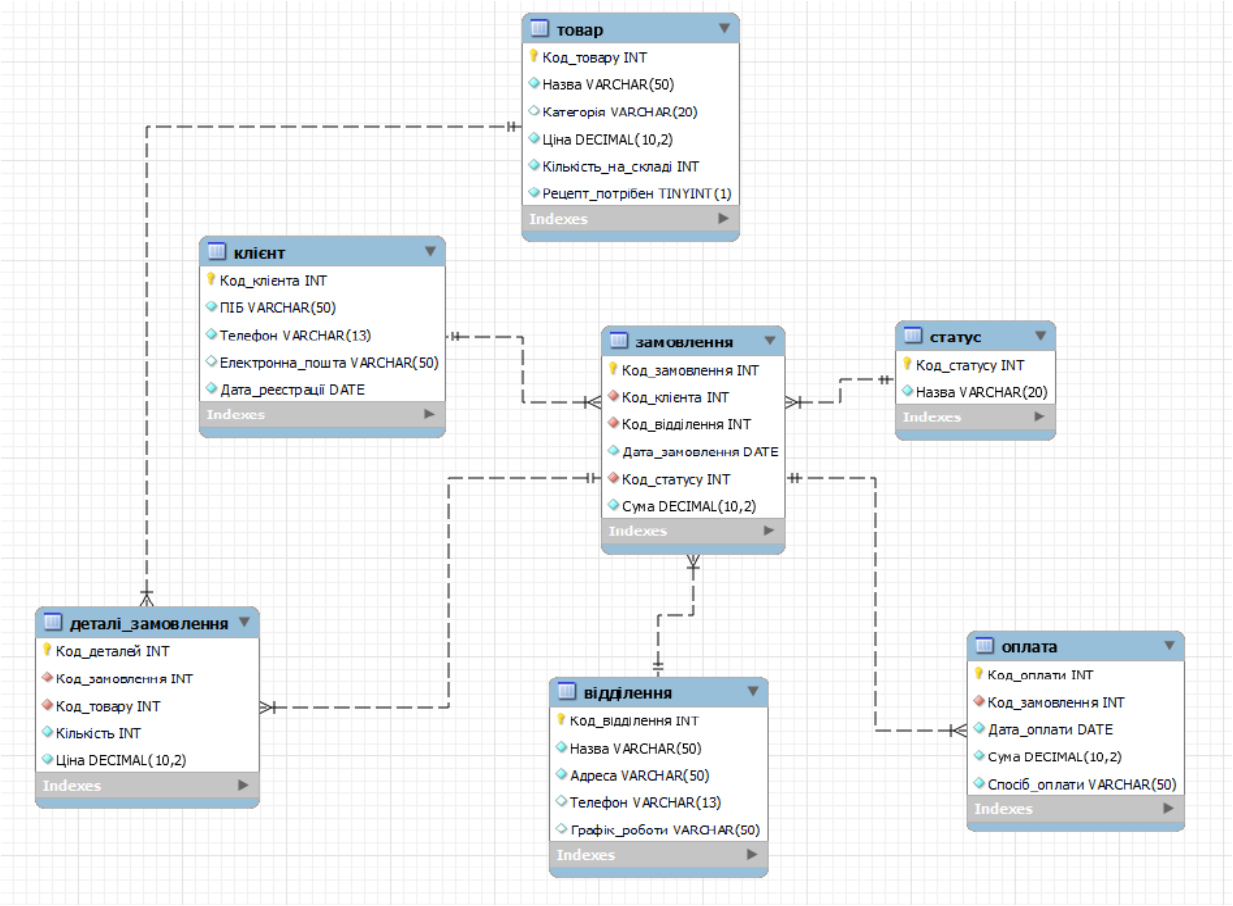

Рис. 3.3. Структура та зв’язки

Таблиця **Клієнт** містить інформацію про клієнтів (табл. 3.1). Поле **Код_клієнта** визначено як INT AUTO_INCREMENT і є первинним ключем (PRIMARY KEY), що гарантує унікальність запису. Поле **ПІБ** має тип VARCHAR(20) і є обов’язковим (NOT NULL), забезпечуючи збереження імен клієнтів. Поле **Телефон**, типу VARCHAR(13), також є обов’язковим і дозволяє зберігати номери телефонів. Поле **Електронна_пошта** має тип VARCHAR(50) і є необов’язковим (NULL), зберігаючи електронну пошту

клієнта за потреби. Поле **Дата_реєстрації**, типу DATE, є обов'язковим для заповнення.

Таблиця 3.1

Структура таблиці Клієнт

Назва поля	Тип даних	Обмеження	Опис
Код_клієнта	INT	PRIMARY KEY, AUTO_INCREMENT, NOT NULL	Унікальний ідентифікатор клієнта
ПІБ	VARCHAR(20)	NOT NULL	Повне ім'я клієнта
Телефон	VARCHAR(13)	NOT NULL	Контактний номер телефону
Електронна_пошта	VARCHAR(50)	NULL	Електронна адреса клієнта (за потреби)
Дата_реєстрації	DATE	NOT NULL	Дата реєстрації клієнта

Таблиця **Відділення** описує інформацію про філії (табл. 3.2). Поле **Код_відділення** є INT AUTO_INCREMENT і виступає первинним ключем (PRIMARY KEY). Поле **Назва**, типу VARCHAR(50), є обов'язковим і використовується для збереження назви відділення.

Таблиця 3.2

Структура таблиці Відділення

Назва поля	Тип даних	Обмеження	Опис
Код_відділення	INT	PRIMARY KEY, AUTO_INCREMENT, NOT NULL	Унікальний ідентифікатор відділення

Продовження табл. 3.2

Назва	VARCHAR(50)	NOT NULL	Назва відділення
Адреса	VARCHAR(50)	NOT NULL	Адреса відділення
Телефон	VARCHAR(13)	NULL	Контактний номер (за потреби)
Графік_роботи	VARCHAR(50)	NULL	Графік роботи відділення (за потреби)

Адреса філії зберігається у полі **Адреса**, яке має тип VARCHAR(50) і також є обов'язковим. Поле **Телефон**, типу VARCHAR(13), є необов'язковим і дозволяє зберігати контактний номер. Графік роботи філії можна вказати у полі **Графік_роботи**, типу VARCHAR(50), яке також є необов'язковим.

Таблиця **Статус** зберігає перелік можливих статусів замовлень (табл. 3.3). Поле **Код_статусу** визначено як INT AUTO_INCREMENT і є первинним ключем. Поле **Назва** має тип VARCHAR(20) і є обов'язковим для заповнення, зберігаючи текстову назву статусу.

Таблиця 3.3

Структура таблиці Статус

Назва поля	Тип даних	Обмеження	Опис
Код_статусу	INT	PRIMARY KEY, AUTO_INCREMENT, NOT NULL	Унікальний ідентифікатор статусу
Назва	VARCHAR(20)	NOT NULL	Текстова назва статусу

Таблиця **Товар** містить дані про доступні товари (табл. 3.4). Поле **Код_товару** є INT AUTO_INCREMENT і виступає первинним ключем (PRIMARY KEY). Поле **Назва** має тип VARCHAR(50) і є обов'язковим,

забезпечуючи можливість зберігання назви товару. Категорія товару записується у необов’язковому полі **Категорія**, яке має тип VARCHAR(20). Поле **Ціна** визначено як DECIMAL(10, 2) і є обов’язковим для заповнення, забезпечуючи точність до сотих. Кількість товарів на складі зберігається у полі **Кількість_на_складі**, яке має тип INT і є обов’язковим. Поле **Рецепт_потрібен**, типу BOOLEAN, є обов’язковим і слугує для збереження булевих значень (0 або 1).

Таблиця 3.4

Структура таблиці Товари

Назва поля	Тип даних	Обмеження	Опис
Код_товару	INT	PRIMARY KEY, AUTO_INCREMENT, NOT NULL	Унікальний ідентифікатор товару
Назва	VARCHAR(50)	NOT NULL	Назва товару
Категорія	VARCHAR(20)	NULL	Категорія товару (необов’язкове поле)
Ціна	DECIMAL(10, 2)	NOT NULL	Ціна товару з точністю до сотих
Кількість_на_складі	INT	NOT NULL	Кількість товарів на складі
Рецепт_потрібен	BOOLEAN	NOT NULL	Показує, чи потрібен рецепт (0 або 1)

Таблиця **Замовлення** зберігає інформацію про оформлені замовлення (табл. 3.5). Поле **Код_замовлення** визначено як INT AUTO_INCREMENT і є первинним ключем. Поле **Код_клієнта**, типу INT, є обов’язковим і є зовнішнім ключем (FOREIGN KEY) на таблицю **Клієнт**(**Код_клієнта**). Поле

Код_відділення, також типу INT, є обов’язковим і посилається на таблицю **Відділення**(**Код_відділення**). Поле **Дата_замовлення** має тип DATE і є обов’язковим для збереження дати замовлення. Статус замовлення визначається через обов’язкове поле **Код_статусу**, яке є зовнішнім ключем на таблицю **Статус**(**Код_статусу**). Поле **Сума** має тип DECIMAL(10, 2) і зберігає сумарну вартість замовлення.

Таблиця 3.5

Структура таблиці Замовлення

Назва поля	Тип даних	Обмеження	Опис
Код_замовлення	INT	PRIMARY KEY, AUTO_INCREMENT, NOT NULL	Унікальний ідентифікатор замовлення
Код_клієнта	INT	NOT NULL, FOREIGN KEY (Клієнт)	Ідентифікатор клієнта, посилання на таблицю Клієнт
Код_відділення	INT	NOT NULL, FOREIGN KEY (Відділення)	Ідентифікатор відділення, посилання на таблицю Відділення
Дата_замовлення	DATE	NOT NULL	Дата оформлення замовлення
Код_статусу	INT	NOT NULL, FOREIGN KEY (Статус)	Ідентифікатор статусу замовлення, посилання на таблицю Статус
Сума	DECIMAL(10, 2)	NULL	Загальна вартість замовлення

Таблиця **Деталі_замовлення** описує товари, включені до замовлення (табл. 3.6). Поле **Код_деталей** є INT AUTO_INCREMENT і виступає первинним ключем. Поля **Код_замовлення** та **Код_товару** мають тип INT і є обов’язковими. Вони виступають зовнішніми ключами на таблиці

Замовлення(Код_замовлення) і **Товар(Код_товару)** відповідно. Поле **Кількість**, типу INT, є обов'язковим для заповнення та вказує кількість товару в замовленні. Поле **Ціна**, типу DECIMAL(10, 2), є обов'язковим і зберігає ціну одиниці товару.

Таблиця 3.6

Структура таблиці Деталі_замовлення

Назва поля	Тип даних	Обмеження	Опис
Код_деталей	INT	PRIMARY KEY, AUTO_INCREMENT, NOT NULL	Унікальний ідентифікатор запису деталі замовлення
Код_замовлення	INT	NOT NULL, FOREIGN KEY (Замовлення)	Ідентифікатор замовлення, посилання на таблицю Замовлення
Код_товару	INT	NOT NULL, FOREIGN KEY (Товар)	Ідентифікатор товару, посилання на таблицю Товар
Кількість	INT	NOT NULL	Кількість товару в замовленні
Ціна	DECIMAL(10, 2)	NOT NULL	Ціна одиниці товару

Таблиця **Оплата** містить дані про платежі (табл. 3.7). Поле **Код_оплати** визначено як INT AUTO_INCREMENT і є первинним ключем. Поле **Код_замовлення**, типу INT, є обов'язковим і посилається на таблицю **Замовлення(Код_замовлення)**. Поле **Дата_оплати** має тип DATE і є обов'язковим для фіксації дати здійснення оплати. Поле **Сума**, типу DECIMAL(10, 2), зберігає розмір платежу та є обов'язковим. Спосіб оплати

зберігається у полі **Спосіб_оплати**, типу VARCHAR(50), яке також є обов’язковим для заповнення.

Таблиця 3.7

Структура таблиці Оплата

Назва поля	Тип даних	Обмеження	Опис
Код_оплати	INT	PRIMARY KEY, AUTO_INCREMENT, NOT NULL	Унікальний ідентифікатор платежу
Код_замовлення	INT	NOT NULL, FOREIGN KEY (Замовлення)	Ідентифікатор замовлення, посилання на таблицю Замовлення
Дата_оплати	DATE	NOT NULL	Дата здійснення платежу
Сума	DECIMAL(10, 2)	NOT NULL	Сума платежу
Спосіб_оплати	VARCHAR(50)	NOT NULL	Спосіб оплати (наприклад, картка, готівка)

Зазначена структура бази даних забезпечує цілісність даних, підтримує їх зберігання та обробку.

Висновки до розділу 3

Було розроблено інфологічну модель бази даних, яка забезпечує логічну структуру з урахуванням основних сутностей, їх атрибутів і взаємозв’язків.

Було визначено ключові сутності — клієнти, відділення, товари, замовлення, статуси замовлень, деталі замовлень та оплати. Встановлено зовнішні ключі для підтримки зв’язків між таблицями.

РОЗДІЛ 4
РОЗРОБКА СИСТЕМИ

4.1. Обґрунтування вибору засобів розробки

У межах розробки інтерактивної системи замовлення та видачі ліків, що складається з вебінтерфейсу для клієнтів, десктоп-додатку для аптекарів та графічної 3D-візуалізації медикаментів, необхідно провести аналіз та вибір відповідних інструментів розробки. Враховуючи технічні та функціональні вимоги до системи, було прийнято рішення про використання таких технологій:

- JavaScript
- C++ з використанням OpenGL
- MySQL

JavaScript (вебчастина) — мова програмування, яка стала галузевим стандартом для побудови інтерактивних користувацьких інтерфейсів у веб-середовищі [17]. Застосування JavaScript дозволяє створити динамічний вебзастосунок з підтримкою асинхронної взаємодії з сервером через API, миттєве оновлення контенту, обробку подій користувача та адаптивний дизайн. Додатково буде використано популярний фреймворк React. Такий вибір зумовлено тим, що цей фреймворк значно полегшує супровід та побудову архітектури додатку. Завдяки React JavaScript також можливо забезпечити гнучку інтеграцію із серверною частиною, модулем авторизації, валідацією введених даних і візуальними повідомленнями про статуси замовлення.

C++ з використанням OpenGL (десктоп-додаток) обрано як основну платформу для створення додатку для функції підтвердження видачі замовлень, а також графічну візуалізацію упаковок ліків у 3D. C++ забезпечує низькорівневий контроль над ресурсами системи. Використання OpenGL дозволяє реалізувати візуалізацію полиць з медикаментами, з можливістю перегляду інформації про товар.

База даних MySQL використовується для централізованого зберігання інформації про користувачів, замовлення, наявні медикаменти, статуси обробки та час взаємодії. Реляційна модель даних забезпечує консистентність даних. За допомогою MySQL буде реалізовано механізми зв’язків між таблицями. Доступ до БД з веб-клієнта організовується через RESTful API, щоб чітко розмежувати логіку взаємодії між компонентами.

OpenGL обрано серед інших графічних бібліотек через його широку підтримку в C++-середовищах, кросплатформеність, низький рівень доступу до графічного конвеєра та підтримку сучасних засобів візуалізації. Через OpenGL стає можлива швидка побудова сцен, шейдерна модель, освітлення, текстурування.

Для розробки, налагодження та збірки десктоп-додатку використовується Microsoft Foundation Classes (MFC) — бібліотека класів для створення графічних інтерфейсів користувача на платформі Windows [8]. MFC надає доступ до взаємодії з Windows API, підтримку обробки подій, створення вікон, діалогів та контролів. Таким чином буде реалізований функціонал програми для видачі ліків та у окремому вікні малюватися полиця з медикаментами ліків.

Вибір поділу системи на дві частини — вебклієнт та десктопний застосунок — зумовлений особливостями процесу: клієнтська частина повинна бути доступна з будь-якого пристрою, а службова — тільки на пристрої аптеки. Це дозволяє уникнути перевантаження єдиної системи, розділити зони відповідальності між інтерфейсами та реалізувати спеціалізовану графіку там, де це доцільно.

4.2. Розробка сайту

Сайт створюється для взаємодії з користувачем. За його допомогою клієнт має можливість вибрати ліки, здійснити замовлення, обрати пункт видачі та переглянути статус замовлення [Додаток А].

Головна сторінка — це перше, що бачить відвідувач. Вона виконує роль навігаційного центру та одночасно має задачу за кілька секунд переконати користувача лишитися на сайті. У верхній частині розміщено вітальний блок із великим заголовком «Ласкаво просимо до Онлайн Аптеки», коротким описом послуги та кнопкою переходу до каталогу (Рис. 4.1.).

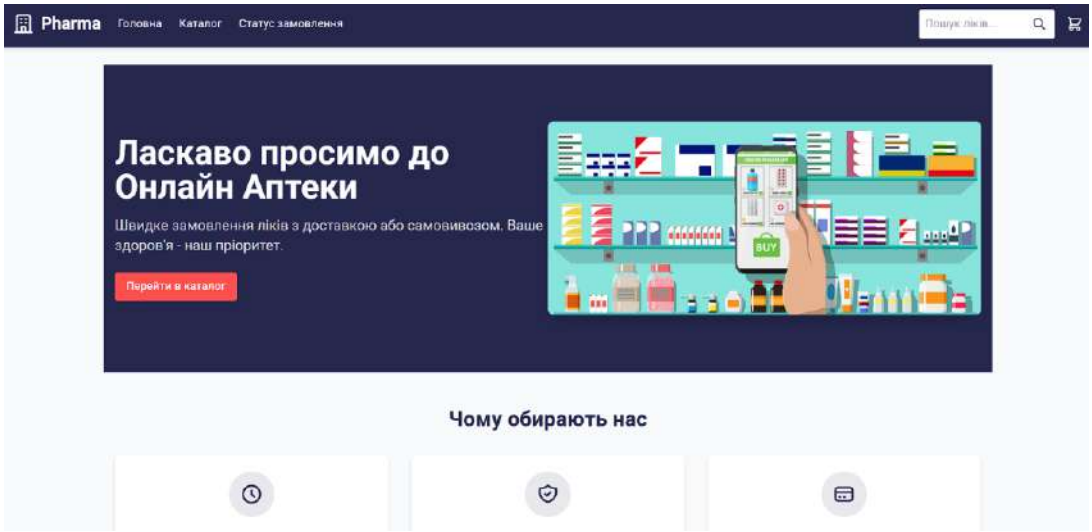

Рис. 4.1. Вітальний банер аптеки

Банер містить один заклик до дії, короткий опис і велике зображення.

Далі йде секція з трьома основними перевагами сервісу (Рис. 4.2.): швидка доставка, гарантія якості та зручна оплата. Кожен блок має іконку, короткий заголовок і пояснення. Вони створені як окремі картки з рівномірним розміщенням, щоб зручно було переглядати на будь-якому пристрої — як на комп’ютері, так і на телефоні (Рис. 4.3.).

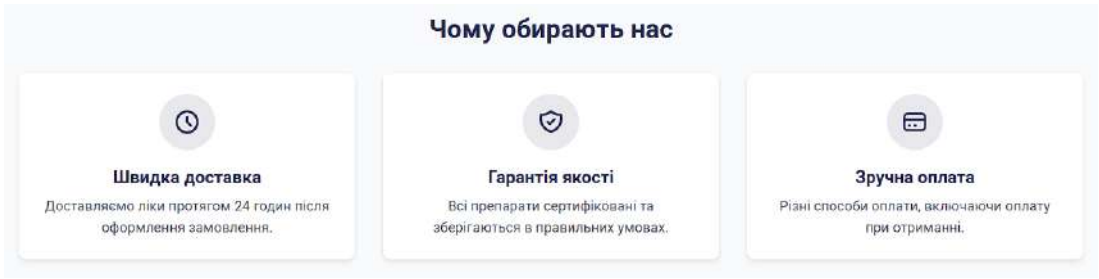

Рис. 4.2. Переваги сервісу десктопна версія

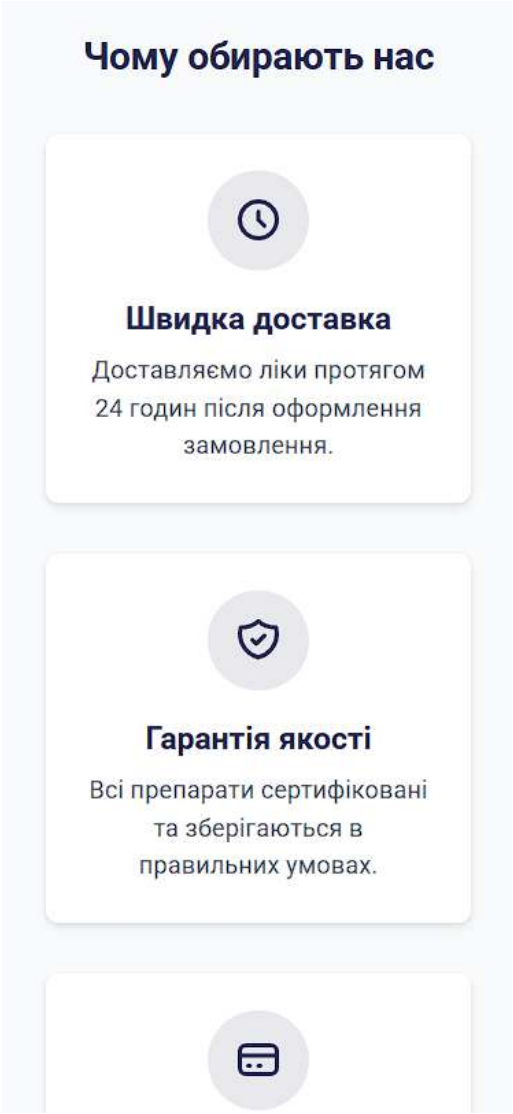

Рис. 4.3. Переваги сервісу мобільна версія

Завершується сторінка ще одним закликом до дії: «Не гайте час — оформіть замовлення онлайн» (Рис. 4.4.). Тут знову є кнопка переходу в каталог і ще один текстовий аргумент про асортимент і ціну. Це потрібно для тих, хто після прокрутки сторінки ще не натиснув на першу кнопку — подвійне нагадування працює на підвищення конверсії.

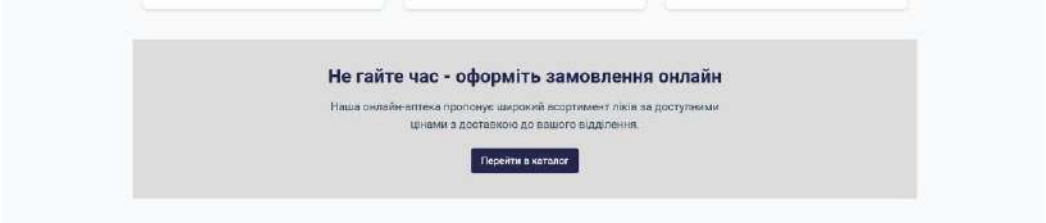

Рис. 4.4. Другий заклик до дії

Хедер сайту містить кілька корисних інтерактивних компонентів (Рис. 4.5.). У верхній частині сайту розташовано логотип з текстовим написом "Pharma", поєднаний із іконкою, який веде користувача на головну сторінку.

Рис. 4.5. Хедер сайту десктопна версія

Праворуч від нього вбудовано основну навігацію — посилання на розділи «Головна», «Каталог» та «Статус замовлення». Вони відображаються лише на більших екранах і зникають у мобільній версії, щоб зберегти компактність інтерфейсу. Для мобільних пристроїв передбачено випадаюче меню. Воно відкривається при натисканні на кнопку-гамбургер і містить ті ж самі навігаційні посилання, що й десктопна версія (Рис. 4.6.).

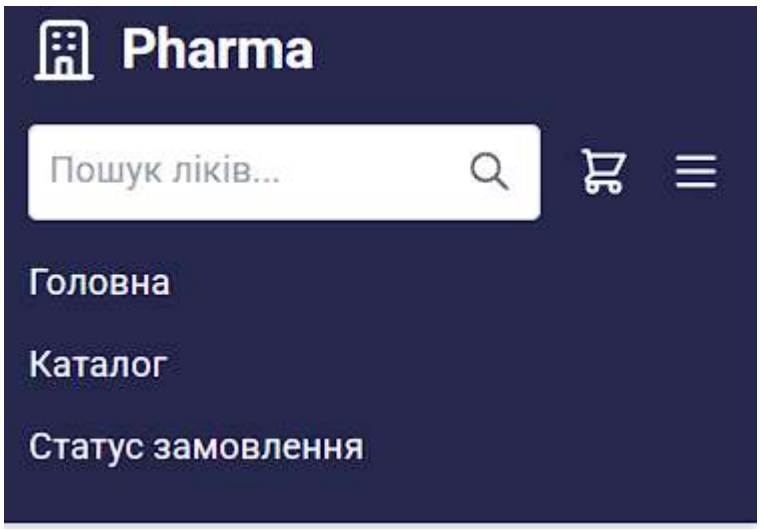

Рис. 4.6. Хедер сайту мобільна версія

Також у хедері є форма пошуку, яка дає змогу шукати ліки за назвою. Коли користувач вводить запит і натискає кнопку пошуку, його автоматично перенаправляє на сторінку каталогу із параметром пошуку, вбудованим у URL. Результати відображаються динамічно (Рис. 4.7.).

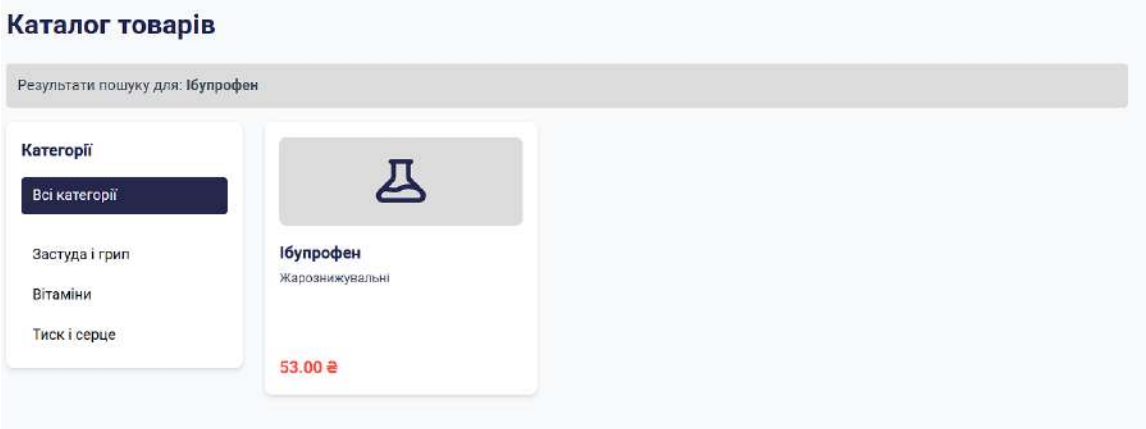

Рис. 4.7. Результати пошуку за запитом

Поруч із формою пошуку розміщено іконку кошика. Вона відображає кількість доданих товарів — числовий індикатор з'являється, лише якщо в кошику щось є (Рис. 4.8.). Іконка клікабельна і веде на сторінку з переліком замовлень.

Рис. 4.8. Індикатор кошика

Футер сайту це завершальний елемент сторінки, але водночас він залишається корисним джерелом навігації та контактної інформації. Візуально він оформлений у темному кольорі фону з білим текстом, що робить його добре помітним і відокремлює від основного контенту сторінки (Рис. 4.9.).

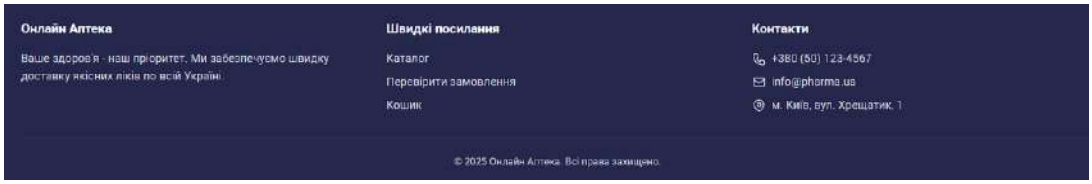

Рис. 4.9. Футер сайту десктопна версія

В лівій частині розміщено короткий опис проекту — невеликий вступ, який нагадує користувачу про головну ідею платформи: швидка та надійна доставка ліків по Україні. Це створює довіру і підкреслює турботу про клієнта.

Центральна частина містить набір посилань, які можна назвати "швидким доступом" до основних сторінок. Тут є переходи на каталог, кошик і перевірку статусу замовлення. Завдяки цьому користувач може швидко

потрапити туди, куди йому найімовірніше потрібно, не повертаючись до верхньої частини сайту.

Праворуч розміщено контактну інформацію: номер телефону, електронна пошта і адреса. Біля кожного пункту є маленька іконка, що робить сприйняття інформації більш наочним. Це все подано в зручному для читання форматі. Нижче футер завершується лінією та приміткою про авторські права — типовий, але потрібний елемент для офіційності ресурсу.

У мобільній версії футер зберігає свою функціональність, але адаптується до компактного формату. Замість трьох колонок, як у десктопній версії, блоки виводяться вертикально один за одним, що дозволяє зручно переглядати контент навіть на вузькому екрані смартфона (Рис. 4.10.). Весь текст залишається добре читабельним, елементи не злипаються, і між ними є достатньо відступів, щоби уникнути випадкових натискань.

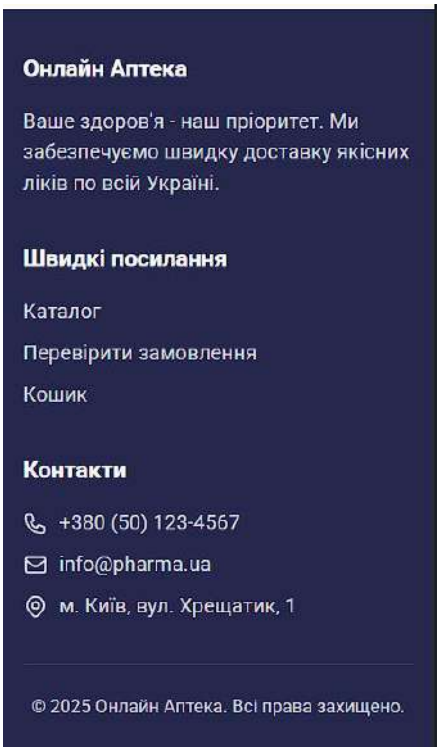

Рис. 4.10. Футер сайту мобільна версія

Іконки біля контактної інформації залишаються помітними, а посилання на основні сторінки легко клікабельні пальцем. Завдяки адаптивній верстці контент футера не обрізається і не виходить за межі екрану.

Каталог містить заголовок з назвою сторінки. Після нього — блок з результатами пошуку, який з’являється лише при активному запиті. Нижче розміщено основну частину, яка на широких екранах відображається у два стовпці. Зліва — вертикальний блок фільтрів. Справа — область з товарами (Рис. 4.11.).

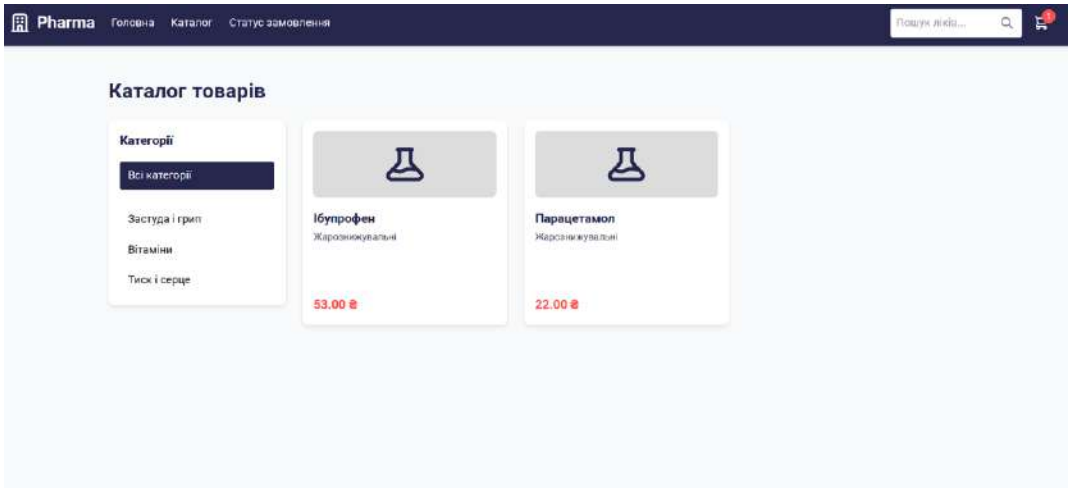

Рис. 4.11. Каталог ліків

У стані завантаження відображається анімований спінер. Якщо товари не знайдено, показується іконка, заголовок і короткий текст з поясненням (Рис. 4.12.). Якщо товари є — вони представлені у вигляді сітки, яка адаптується: один стовпець на маленьких екранах, два — на середніх, три — на великих.

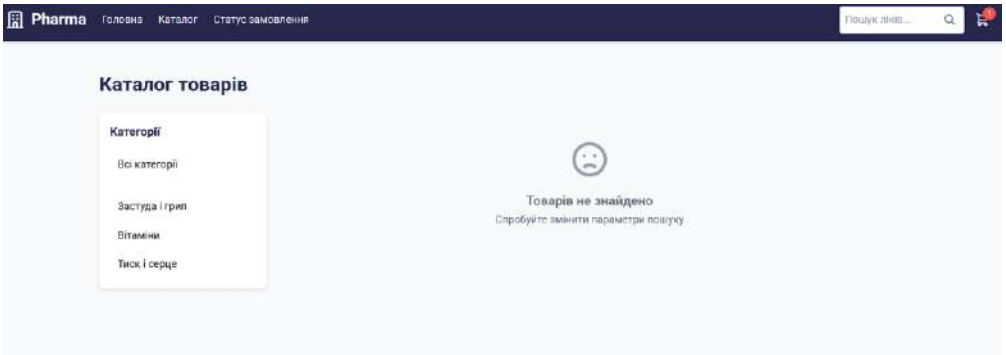

Рис. 4.12. Відсутність товарів у десктопній версії

Кожна товарна картка містить іконку в окремому блоці, назву, категорію, ціну, і нижню панель з кнопкою додавання в кошик. Кнопка містить іконку і текст. Вся картка є посиланням на сторінку конкретного товару.

Фільтри розміщені в бічній панелі, яка на великих екранах має фіксовану ширину, а на мобільних займає всю ширину. Вгорі мобільної версії є заголовок з кнопкою, що відкриває або закриває список фільтрів. При натисканні стрілка анімується, змінюючи напрямок (Рис. 4.13.).

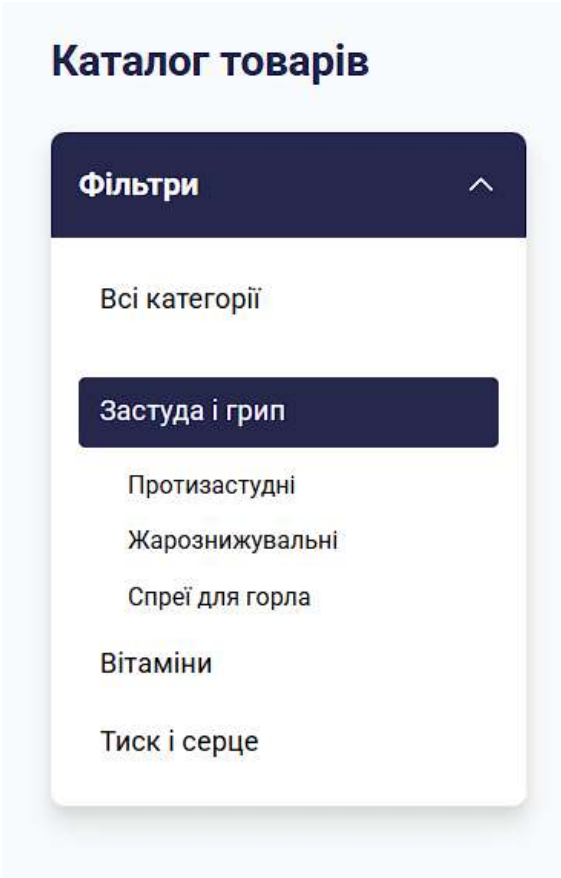

Рис. 4.13. Фільтри у мобільній версії

Після заголовка відкривається область з категоріями. Спершу є кнопка для скидання вибору, яка виділяється, якщо не вибрана жодна категорія. Нижче йде перелік основних категорій, кожна відображається у вигляді кнопки, яка змінює фон і колір при виборі або наведенні.

Якщо категорія вибрана і має підкатегорії, вони розкриваються вкладеним списком із відступом і меншою величиною тексту. Підкатегорії теж представлені у вигляді кнопок з відповідною стилізацією для активного та неактивного стану.

Мобільна версія каталогу має вертикальне розташування елементів. Товари відображаються в один стовпець, кожен займає всю ширину екрану, що робить їх зручними для перегляду на вузьких екранах. Кнопки та елементи

інтерфейсу мають достатній розмір для зручного натискання пальцем (Рис. 4.14.).

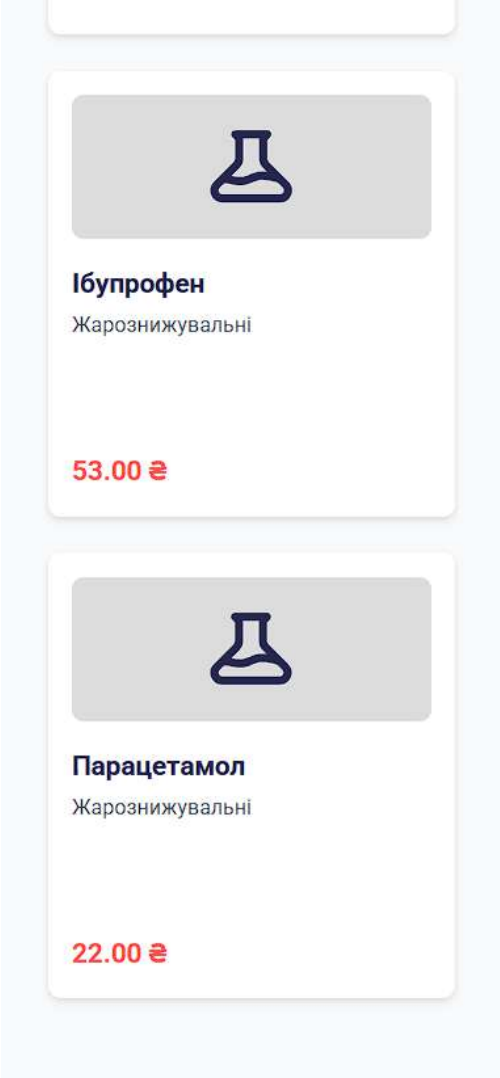

Рис. 4.14. Каталог у мобільній версії

Заголовки, тексти та кнопки адаптуються за розміром шрифтів і відступів, щоб зберегти читабельність і комфортну взаємодію. Завантаження і повідомлення про відсутність товарів займають достатньо місця, щоб бути помітними, але не перевантажувати екран (Рис. 4.15.).

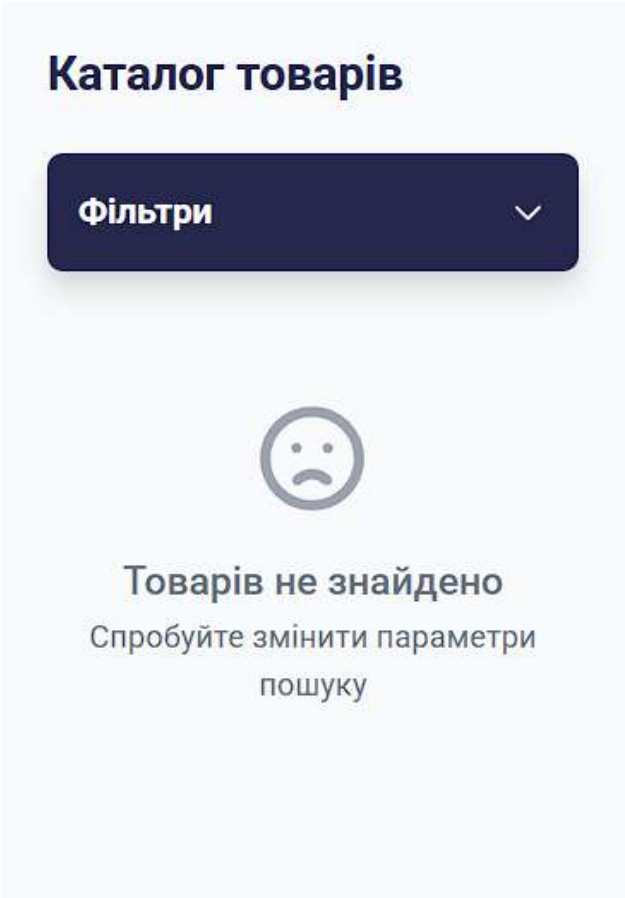

Рис. 4.15. Відсутність товарів у мобільній версії

Сторінка товару складається з контейнера з відступами, що центрує вміст. У верхній частині є посилання назад до каталогу з іконкою стрілки (Рис. 4.16.). Основний блок розбитий на дві колонки (на десктопі): зліва — квадратна область для зображення товару (з іконкою-заглушкою), справа — вся інформація про товар. Заголовок товару великий і жирний, нижче — категорія у вигляді бейджу, а також статус наявності, якщо є. Далі опис товару у вигляді параграфа, а потім велика ціна, виділена кольором.

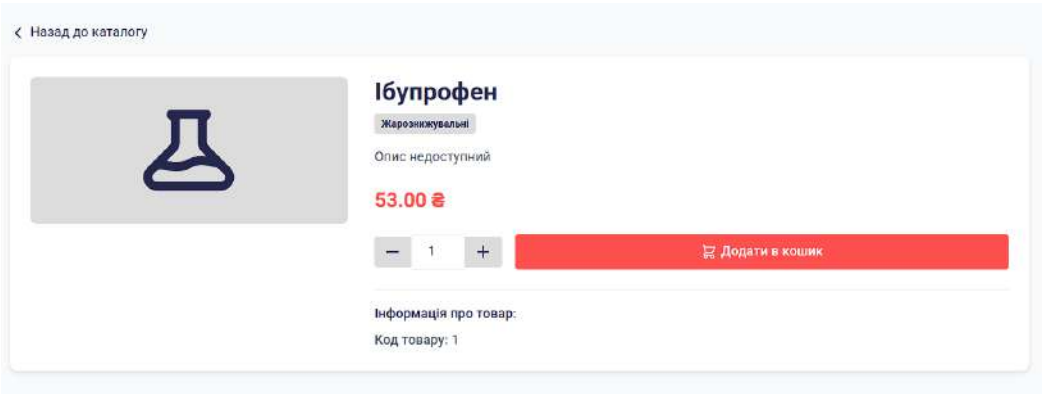

Рис. 4.16. Сторінка товару у десктопній версії

Під ціною — контрол для вибору кількості (кнопки мінус і плюс по боках, між ними числове поле з центруванням) та кнопка додавання до кошика. Кнопка змінює вигляд і текст після додавання (Рис. 4.17.). Внизу — секція з інформацією про товар (код, виробник, країна виробництва) у вигляді списку. У разі завантаження показується анімований індикатор. Якщо товар не знайдено — текст з посиланням назад у каталог.

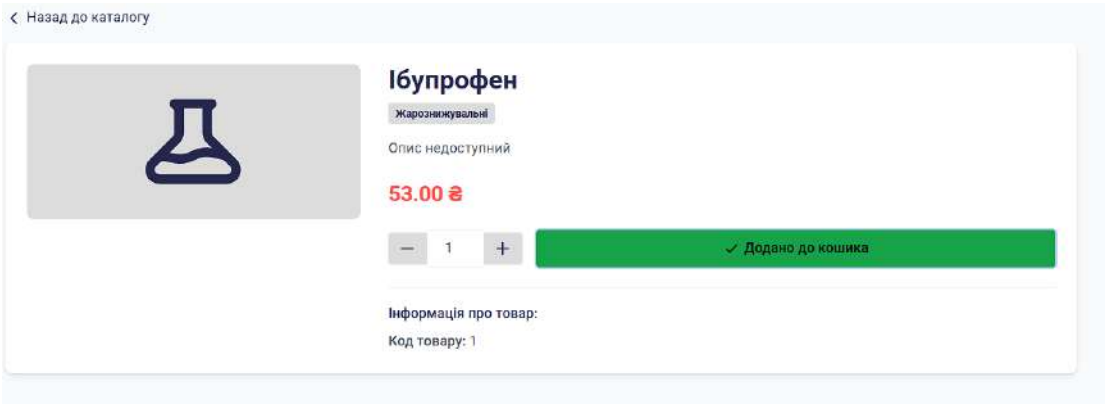

Рис. 4.17. Зміна сторінки товару, після додавання товару у кошик

На мобільних пристроях колонки переходять у вертикальний стовпець, зображення і текст розташовані послідовно (Рис. 4.18.).

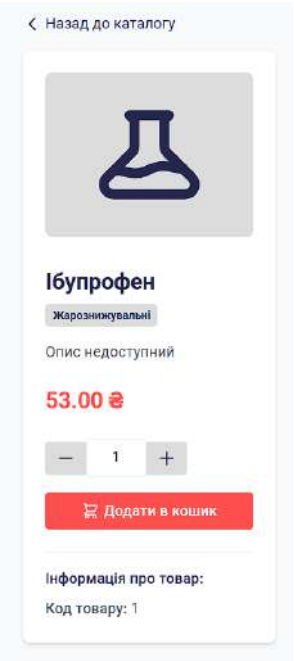

Рис. 4.18. Сторінка товару у мобільній версії

Кошик — це сторінка, яка служить для перегляду та управління товарами, які користувач додав для подальшої покупки. На цій сторінці

відразу видно, які позиції були обрані, скільки кожного товару, а також їхню загальну вартість. За цей функціонал також відповідає серверна чатсина сайту [Додаток Б].

Якщо користувач ще не додав жодного товару, замість звичного вмісту з’являється повідомлення про те, що кошик порожній (Рис. 4.19.).

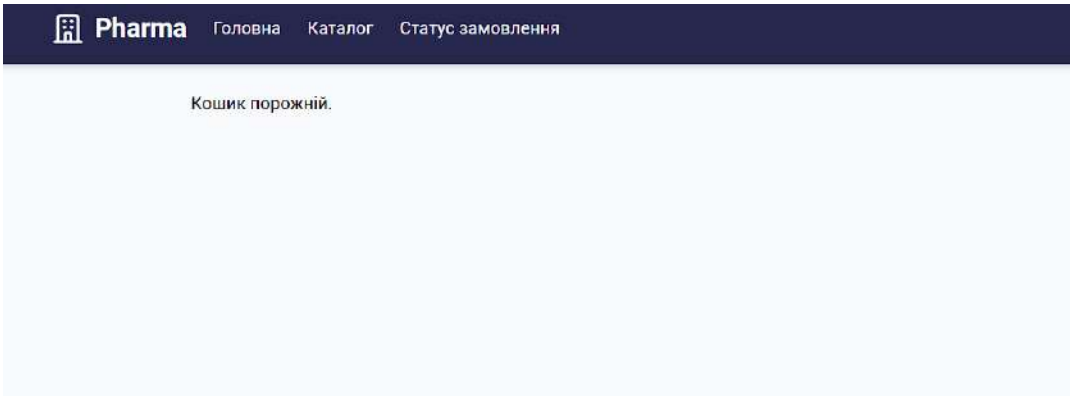

Рис. 4.19. Порожній кошик

Якщо в кошику є товари, кожен з них відображається у вигляді окремої картки з назвою, кількістю та ціною за одиницю, а також загальною вартістю цієї позиції (Рис. 4.20.). Інформація подана таким чином, щоб користувач міг швидко зрозуміти, що він обрав і скільки це коштує. Поруч із кожним товаром розташована кнопка видалення, яка може швидко прибрати товар із кошика.

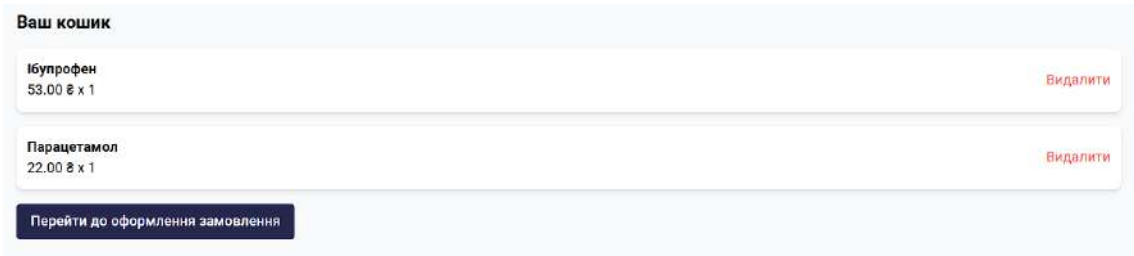

Рис. 4.20. Кошик з товарами

Нижче розташована кнопка для переходу до оформлення замовлення. Ця кнопка веде до сторінки підтвердження замовлення. Сторінка підтвердження замовлення — це фінальний крок у процесі покупки, де користувач вводить свої контактні дані, вибирає зручне відділення для отримання товару та переглядає список обраних товарів перед остаточним підтвердженням замовлення через спеціальну форму (Рис. 4.21.).

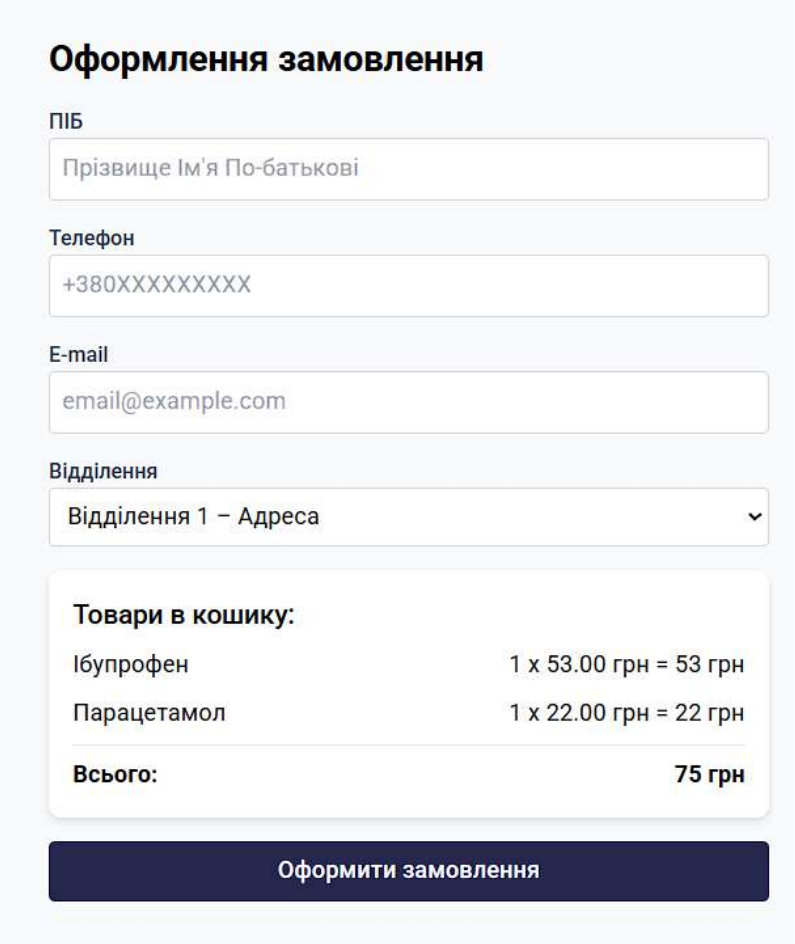

Рис. 4.21. Форма оформлення замовдення

Першим полем у формі є текстовий інпут для введення ПІБ. Поруч або над полем є напис «ПІБ», який зв’язаний з цим інпутом. У самому полі відображається підказка, що пропонує ввести прізвище, ім’я та по батькові. Якщо поле залишити порожнім або некоректно заповнити, під інпутом з’явиться повідомлення про помилку червоним кольором.

Наступним полем є інпут для номера телефону. Йому передує підпис «Телефон». В полі є плейсхолдер із прикладом українського номера телефону. Це обов’язкове поле, і у разі помилки введення користувач також побачить червоне повідомлення.

Далі йде інпут для електронної пошти, підписаний як «E-mail». Це поле не є обов’язковим, але якщо воно заповнене, має пройти перевірку на правильність формату адреси. Під інпутом у разі помилки валідації показується відповідне повідомлення.

Нижче розташований випадаючий список для вибору відділення. Він має підпис «Відділення». Спочатку вибрано перше відділення зі списку, отриманого з сервера. Якщо користувач не вибере жодного варіанту, система покаже помилку, яку відобразить червоним текстом під списком.

Під формою відображається блок із товарами, які користувач додав у кошик. Для кожного товару показується назва, кількість, ціна за одиницю та загальна вартість. Наприкінці цього блоку показується підсумкова сума замовлення.

Кнопка для підтвердження замовлення розміщена внизу форми і займає всю ширину. Якщо форма зараз відправляється, на кнопці відображається текст «Обробка...», і кнопка вимкнена. Якщо кошик порожній, кнопка також недоступна.

Після спроби відправлення форми під кнопкою з’являється повідомлення про результат: зелений блок з підтвердженням успішного замовлення (Рис. 4.22.) або червоний — із повідомленням про помилку.

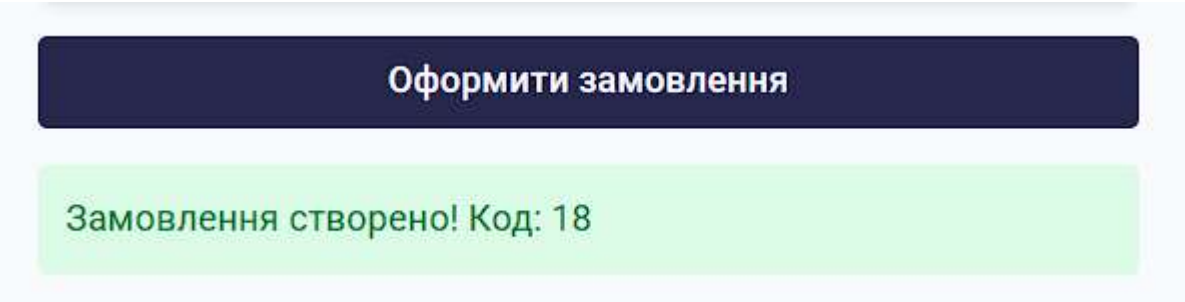

Рис. 4.22. Створення замовлення

Сторінка деталей замовлення дозволяє користувачу перевірити статус свого замовлення за кодом. Починається вона з кількох станів: для збереження введеного коду замовлення, отриманої інформації про замовлення та повідомлення про помилку.

Інтерфейс містить форму, де користувач вводить номер замовлення у поле з типом "number". Це поле має стиль "input", має заповнюватися обов’язково і підтримує зміну значення через onChange, оновлюючи відповідний стан. Під полем розташована кнопка для відправлення форми,

оформлена як кнопка з класами "btn btn-primary" і розтягнута на всю ширину контейнера.

При відправленні форми запускається функція, яка запобігає стандартній поведінці браузера, очищує повідомлення про помилку та попередні дані замовлення, потім робить запит на сервер за детальною інформацією про замовлення по введеному коду.

Якщо замовлення знайдено, його дані зберігаються у стані і виводяться на сторінку у спеціальному блоці з класами, які додають відступи і стилі карти (Рис. 4.23.). У цьому блоці заголовок з текстом "Інформація про замовлення" виконаний жирним шрифтом трохи більшого розміру. Далі відображаються ключові дані замовлення: код замовлення, дата, статус у вигляді помітного кольорового бейджу (синій фон і білий текст), а також загальна сума в гривнях.

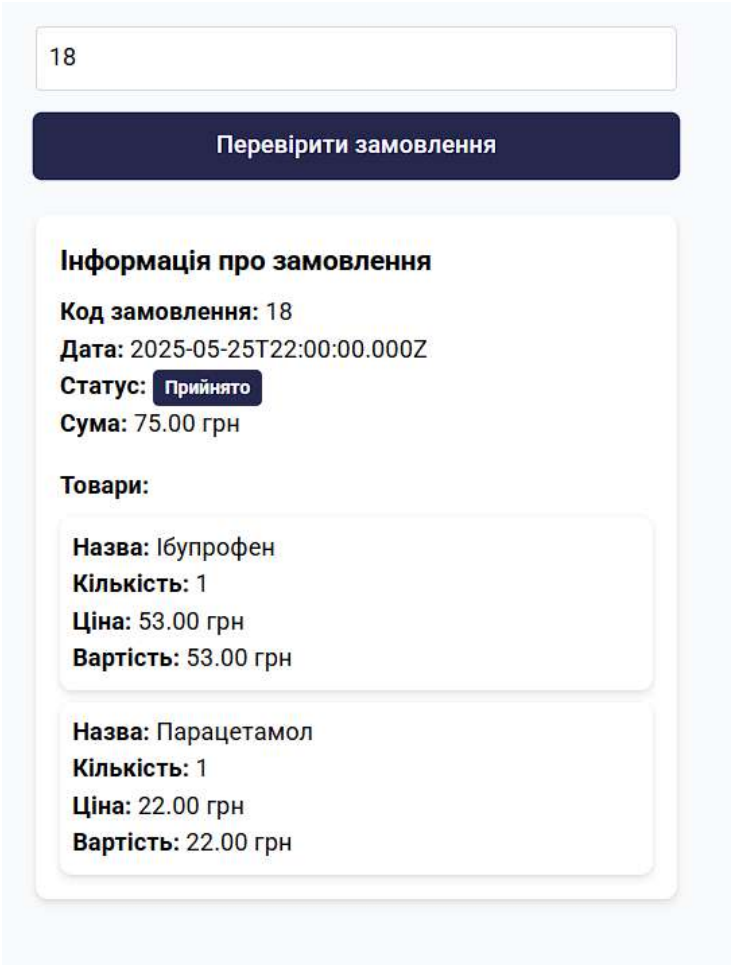

Рис. 4.23. Перевірка замовлення

Під основною інформацією є заголовок, що відокремлює список товарів у замовленні. Кожен товар показується у своєму відокремленому блоці,

стилізованому як картка з внутрішніми відступами. Для кожного товару наведені деталі: назва, кількість, ціна за одиницю та загальна вартість, усе оформлено з використанням жирного виділення для підписів.

Якщо під час запиту виникає помилка або замовлення з таким кодом не знайдено, внизу сторінки з'являється текстове повідомлення у вигляді параграфа із вказівкою на проблему (Рис. 4.24.).

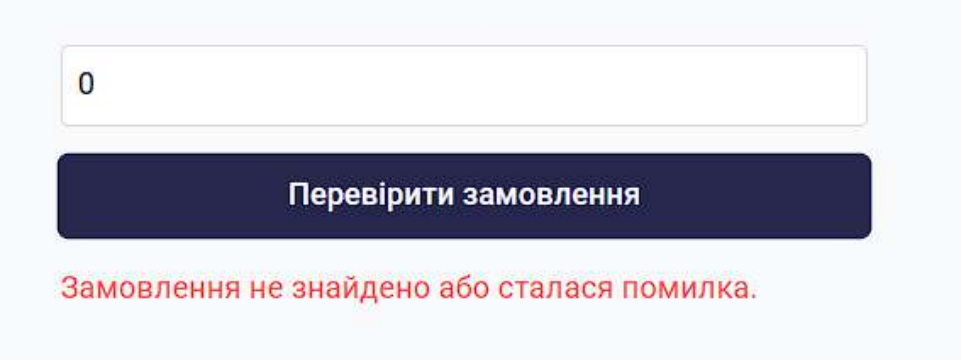

Рис. 4.24. Помилка пошуку замовлення

Весь контент обгорнутий у контейнер із максимальною шириною, центрується горизонтально і має внутрішні відступи для комфортного читання.

4.3. Розробка десктопного додатку

Мета створення додатку — це надати працівникам аптеки інструмент, який допоможе організувати роботу із замовленнями: переглядати актуальні дані, контролювати їх статуси, вносити зміни щодо доставки, а також забезпечувати точний облік замовлень і пов'язаних з ними даних.

Вікно під назвою "Перегляд і керування замовленнями" служить основним інтерфейсом для роботи з замовленнями у системі аптеки. Воно призначене для відображення повного списку замовлень, що дозволяє працівникам швидко отримати інформацію про кожне замовлення, а також виконувати операції зі зміни їх статусів (Рис. 4.25.).

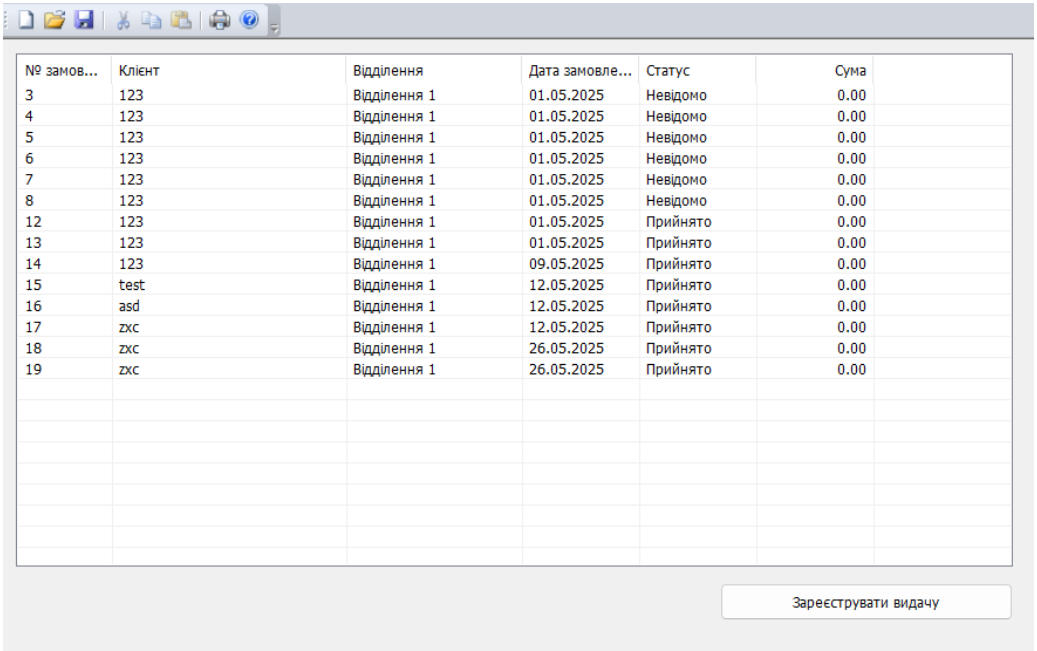

Рис. 4.25. Перегляд замовлень

Інтерфейс вікна складається з таблиці, у якій кожен рядок відображає одне замовлення. У таблиці є шість колонок: номер замовлення, ім’я клієнта, відділення аптеки, дата замовлення, статус замовлення та загальна сума. Ці колонки допомагають одразу побачити найважливіші деталі, необхідні для розуміння поточного стану замовлення.

Таблиця підтримує вибір рядка, що дає змогу користувачу виділити конкретне замовлення для подальшої роботи з ним. Під таблицею розташована кнопка для реєстрації доставки, яка відкриває діалогове вікно. У цьому вікні можна позначити статус замовлення як доставлене або в процесі відправки (Рис. 4.26.).

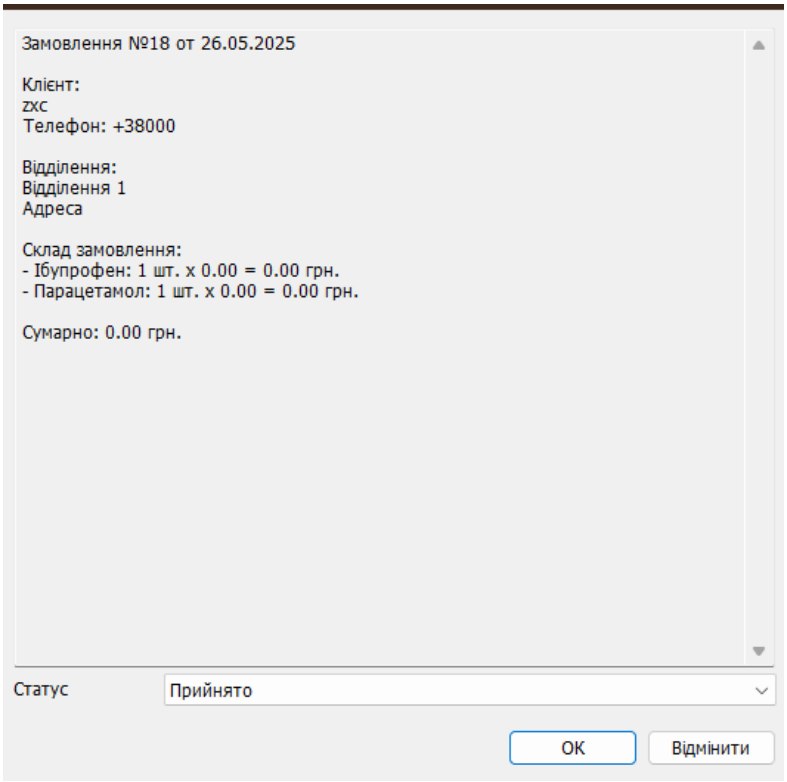

Рис. 4.26. Зміна статусу замовлення

Якщо ж користувач намагається натиснути цю кнопку, не вибравши замовлення, система видає повідомлення з нагадуванням про необхідність вибору (Рис. 4.27.).

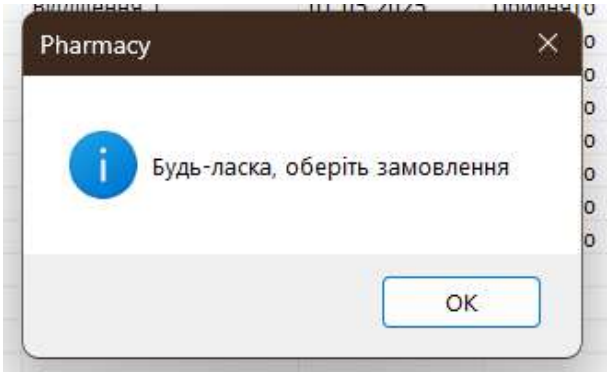

Рис. 4.27. Зміна статусу без вибору замовлення

Візуально таблиця налаштована так, щоб було зручно виділяти весь рядок при кліку і щоб між рядками було видно сітку, що допомагає краще орієнтуватись у великому обсязі інформації. Крім того, при натисканні правою кнопкою миші відкривається контекстне меню, яке надає додаткові можливості для роботи з обраним замовленням.

Вікно адаптується до розміру батьківського вікна, підтримуючи коректне відображення елементів без зайвих прокруток. При завантаженні

вікна відбувається ініціалізація таблиці і заповнення її даними, отриманими з документу програми, що зберігає інформацію про всі замовлення, клієнтів, статуси і відділення.

Користувач потрапляє у вікно Візуальної полиці через головне меню, обравши розділ "Візуалізація" або натиснувши відповідну кнопку на панелі інструментів. Це вікно є інтерфейсом для відображення та контролю стану асортименту товару в аптеці.

Вікно Візуальної полиці займає більшу частину екрану і поділене на кілька зон (Рис. 4.28.). Основна зона — це площа з візуальним представленням замовлень у вигляді плиток або карток, які групуються за статусами або етапами виконання. Кожна плитка містить інформацію про замовлення: його номер, клієнта, дату створення та поточний статус. Кольорове кодування та іконки допомагають швидко орієнтуватися у стані кожного замовлення.

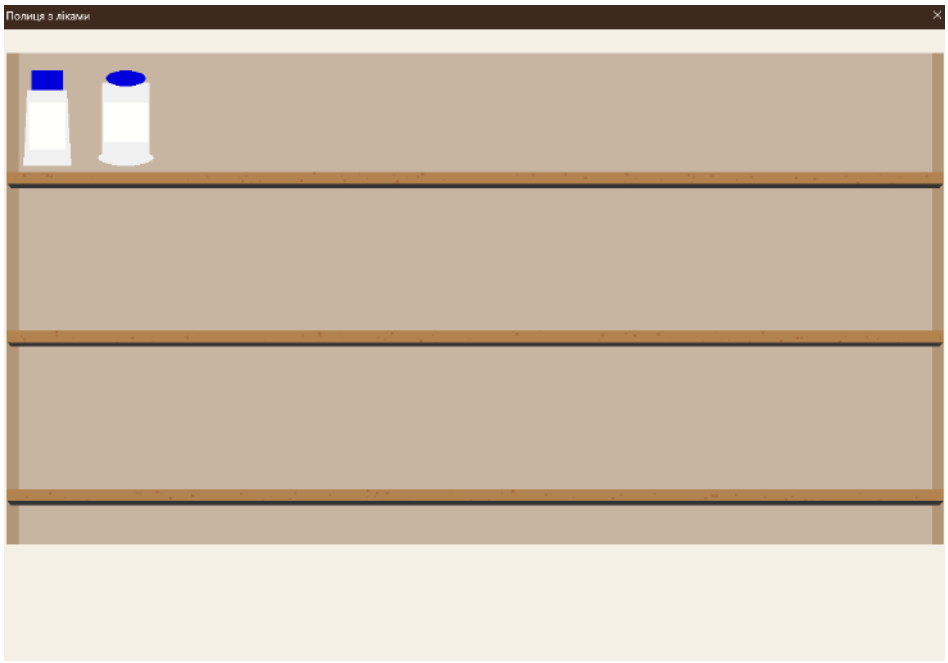

Рис. 4.28. Візуальна полиця з ліками

Користувач може взаємодіяти з плитками ліків — натискаючи на них, він відкриває детальну інформацію про конкретний препарат, де відображаються його назва, виробник, кількість на полиці, термін придатності та ціна (Рис. 4.29.).

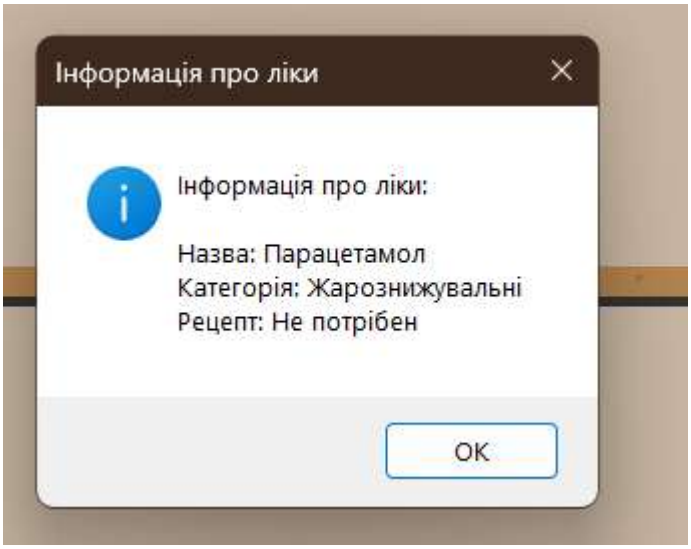

Рис. 4.29. Взаємодія з плитками ліків

Вікно Візуальної полиці дає змогу працівнику контролювати наявність, розміщення та стан ліків у аптеці.

Висновки до розділу 4

Було описано програмні продукти видачі та замовлення ліків: вебсайту для клієнтів, десктоп-додатку для аптекарів та модулів для обробки замовлень і видачі ліків. Описано архітектуру компонентів, алгоритми їх роботи та взаємодію між ними. Реалізовано інтерфейси, що забезпечують пошук, оформлення замовлень і контроль наявності ліків. Запроваджені механізми підтримують обробку даних у реальному часі та збереження інформації у базі даних. Розроблені продукти відповідають технічним вимогам і структурі системи, що була визначена на попередніх етапах.

ВИСНОВКИ

У рамках дипломної роботи було здійснено повний цикл проєктування, розробки та впровадження інтерактивної системи замовлення та видачі лікарських засобів, що передбачає використання графічної візуалізації для підвищення ефективності аптечного обслуговування.

У першому розділі було сформульовано мету та задачі дослідження, обґрунтовано актуальність теми в умовах сучасної цифровізації фармацевтичної сфери. Проведено детальний огляд існуючих веб-рішень у даній галузі, що дозволило виявити їхні обмеження та окреслити напрямки удосконалення. На основі аналізу було сформовано вимоги до майбутньої системи з урахуванням потреб як кінцевих користувачів, так і працівників аптек.

У другому розділі розглянуто проєктування системи. Описано загальну архітектуру, що передбачає поділ на клієнтську та серверну частини, інтеграцію з базою даних, а також візуальні компоненти інтерфейсу. Запропоновано алгоритм функціонування системи, що охоплює процеси від створення замовлення до його обробки та видачі, з акцентом на зручність користувача та мінімізацію часу обслуговування.

У третьому розділі розроблено концептуальну модель бази даних, що відповідає специфіці предметної області. Було створено повноцінну структуру таблиць, визначено взаємозв’язки між ними, а також реалізовано фізичну базу даних з урахуванням вимог до надійності, цілісності та продуктивності зберігання інформації.

У четвертому розділі було обґрунтовано вибір інструментів для реалізації програмного продукту. Створено веб-сайт, що реалізує функціонал онлайн-замовлення медикаментів, а також розроблено десктопний додаток із візуальною полицею — інтерфейсом для контролю за поточним станом асортименту та замовлень. Інтерфейс побудовано відповідно до принципів UX/UI-дизайну, що дозволяє користувачеві інтуїтивно взаємодіяти з системою без необхідності спеціального навчання.

У результаті виконання дипломної роботи було досягнуто такі ключові результати:

Розроблено повноцінну програмну систему, яка охоплює фронт-енд, бек-енд, базу даних і графічну частину інтерфейсу.

Впроваджено механізми, які дозволяють контролювати наявність ліків у візуальному режимі та відслідковувати статус замовлення в режимі реального часу.

Досягнуто значного підвищення зручності взаємодії з аптечною системою для обох сторін — як працівників аптек, так і клієнтів.

Система є масштабованою, її можна адаптувати до потреб аптечних мереж різного розміру, інтегрувати з іншими сервісами (CRM, складський облік, аналітика).

Запропоноване рішення має потенціал для подальшого розвитку — зокрема, впровадження мобільної версії, підтримки електронних рецептів, аналітичного модуля для прогнозування попиту.

Таким чином, поставлена мета дипломної роботи була повністю досягнута. Розроблена система є актуальною, ефективною та має значну практичну цінність. Її впровадження може суттєво підвищити рівень автоматизації та якості обслуговування в аптечній галузі, що відповідає сучасним вимогам до цифрової трансформації медичної інфраструктури.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Elmasri R., Navathe S. Fundamentals of Database Systems / Elmasri R., Navathe S. – Pearson, 2015.
2. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts / Silberschatz A., Korth H. F., Sudarshan S. – McGraw-Hill, 2019.
3. Date C. J. An Introduction to Database Systems / Date C. J. – Addison-Wesley, 2003.
4. Coronel C., Morris S. Database Systems: Design, Implementation, & Management / Coronel C., Morris S. – Cengage Learning, 2019.
5. Pressman R. S. Software Engineering: A Practitioner’s Approach / Pressman R. S. – McGraw-Hill, 2014.
6. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software / Gamma E., Helm R., Johnson R., Vlissides J. – Addison-Wesley, 1994.
7. Kernighan B. W., Ritchie D. M. The C Programming Language / Kernighan B. W., Ritchie D. M. – Prentice Hall, 1988.
8. Microsoft Docs. MFC (Microsoft Foundation Class) Library Documentation [Електронний ресурс]. – Режим доступу : https://learn.microsoft.com/en-us/cpp/mfc/
9. Microsoft Docs. CMake Documentation [Електронний ресурс]. – Режим доступу : https://cmake.org/documentation
10. Fowler M. Patterns of Enterprise Application Architecture / Fowler M. – Addison-Wesley, 2002.
11. Beck K. Test-Driven Development: By Example / Beck K. – Addison-Wesley, 2003.
12. Kent B. Clean Code: A Handbook of Agile Software Craftsmanship / Kent B. – Prentice Hall, 2008.
13. Sommerville I. Software Engineering / Sommerville I. – Pearson, 2015.

14. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.
15. Flanagan D. JavaScript: The Definitive Guide / Flanagan D. – O'Reilly Media, 2020.
16. Robbins J. N. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics / Robbins J. N. – O'Reilly Media, 2018.
17. React Documentation [Електронний ресурс]. – Режим доступу : https://reactjs.org/docs/getting-started.html
18. Node.js Documentation [Електронний ресурс]. – Режим доступу : https://nodejs.org/en/docs/
19. Accomazzo B., Murray M., Hernandez S. Fullstack React: The Complete Guide to ReactJS and Friends / Accomazzo B., Murray M., Hernandez S.
20. Casciaro M., Mammino L. Node.js Design Patterns / Casciaro M., Mammino L.
21. Chinnathambi K. Learning React: Functional Web Development with React and Redux / Chinnathambi K.
22. Cantelon M. et al. Node.js in Action / Cantelon M. et al.
23. Express.js Documentation [Електронний ресурс]. – Режим доступу : https://expressjs.com/

ДОДАТКИ

Додаток А

Код backend-частини сайту

```
const express = require('express');
const router = express.Router();
const pool = require('../db');

router.get('/', async (req, res) => {
  try {
    const [rows] = await pool.query('SELECT * FROM Відділення');
    res.json(rows);
  } catch (err) {
    console.error(err);
    res.status(500).json({    error:    'Помилка    завантаження
відділень' });
  }
});

module.exports = router;

const express = require('express');
const router = express.Router();
const db = require('../db');

router.post('/', async (req, res) => {
  const  {  ПІБ,  Телефон,  Електронна_пошта,  Дата_реєстрації  }  =
req.body;

  try {
    const [result] = await db.query(
      'INSERT  INTO  Клієнт  (ПІБ,  Телефон,  Електронна_пошта,
Дата_реєстрації) VALUES (?, ?, ?, ?)',
      [ПІБ, Телефон, Електронна_пошта, Дата_реєстрації]
    );

    res.json({ success: true, clientId: result.insertId });
  } catch (error) {
    console.error('Помилка при створенні клієнта:', error);
    res.status(500).json({ success: false });
  }
});

module.exports = router;

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

```
const router = express.Router();
const pool = require('../db');

router.get('/:id', async (req, res) => {
    try {
      const { id } = req.params;

      const [[order]] = await pool.query(
        `SELECT
           z.Код_замовлення,
           z.Дата_замовлення,
           s.Назва AS Статус,
           z.Сума
         FROM Замовлення z
         JOIN Статус s ON z.Код_статусу = s.Код_статусу
         WHERE z.Код_замовлення = ?`,
        [id]
      );

      if (!order) {
        return res.status(404).json({ error: 'Замовлення не
знайдено' });
      }

      console.log('ID у запиті:', id, typeof id);

      const [items] = await pool.query(
        `SELECT
           t.Назва,
           d.Кількість,
           d.Ціна,
           (d.Кількість * d.Ціна) AS Вартість
         FROM Деталі_замовлення d
         JOIN Товар t ON d.Код_товару = t.Код_товару
         WHERE d.Код_замовлення = ?`,
        [id]
      );

      res.json({ ...order, Товари: items });
    } catch (err) {
      console.error(err);
      res.status(500).json({ error: 'Помилка отримання
замовлення' });
    }
  });
```

```
  module.exports = router;

const express = require('express');
const router = express.Router();
const pool = require('../db');

router.post('/', async (req, res) => {
  try {
    const { ПІБ, Телефон, Електронна_пошта, Відділення, Товари }
= req.body;

    if (!ПІБ || !Телефон || !Відділення || !Array.isArray(Товари)
|| Товари.length === 0) {
      return res.status(400).json({
        error: 'Неповні дані',
        details: 'Будь ласка, надайте ПІБ, Телефон, Відділення та
Товари'
      });
    }

    const connection = await pool.getConnection();
    await connection.beginTransaction();

    try {
      const [existingClients] = await connection.query(
        'SELECT Код_клієнта FROM Клієнт WHERE Телефон = ?',
        [Телефон]
      );

      let Код_клієнта;

      if (existingClients.length === 0) {
        const [clientResult] = await connection.query(
          'INSERT INTO Клієнт (ПІБ, Телефон, Електронна_пошта,
Дата_реєстрації) VALUES (?, ?, ?, CURDATE())',
          [ПІБ, Телефон, Електронна_пошта]
        );
        Код_клієнта = clientResult.insertId;
      } else {
        Код_клієнта = existingClients[0].Код_клієнта;
      }

      let totalAmount = 0;
```

```
      for (const item of Товари) {
        const Код_товару = parseInt(item.Код_товару, 10);
        const Кількість = parseInt(item.Кількість, 10);

        const [товарResult] = await connection.query(
          'SELECT Ціна, Кількість_на_складі FROM Товар WHERE
Код_товару = ?',
          [Код_товару]
        );

        if (товарResult.length === 0) {
          throw new Error(`Товар з кодом ${Код_товару} не
знайдено`);
        }

        const товар = товарResult[0];

        if (товар.Кількість_на_складі < Кількість) {
          throw new Error(`Недостатня кількість товару з кодом
${Код_товару}. Наявно: ${товар.Кількість_на_складі}, потрібно:
${Кількість}`);
        }

        totalAmount += товар.Ціна * Кількість;
      }

      const [orderResult] = await connection.query(
        'INSERT INTO Замовлення (Код_клієнта, Код_відділення,
Дата_замовлення, Код_статусу, Сума) VALUES (?, ?, CURDATE(), 1,
?)',
        [Код_клієнта, Відділення, totalAmount]
      );

      const Код_замовлення = orderResult.insertId;

      for (const item of Товари) {
        const Код_товару = parseInt(item.Код_товару, 10);
        const Кількість = parseInt(item.Кількість, 10);

        const [товарResult] = await connection.query(
          'SELECT Ціна FROM Товар WHERE Код_товару = ?',
          [Код_товару]
        );

        const Ціна = товарResult[0].Ціна;
```

```
        await connection.query(
          'INSERT    INTO    Деталі_замовлення    (Код_замовлення,
Код_товару, Кількість, Ціна) VALUES (?, ?, ?, ?)',
          [Код_замовлення, Код_товару, Кількість, Ціна]
        );

        await connection.query(
          'UPDATE     Товар     SET     Кількість_на_складі     =
Кількість_на_складі - ? WHERE Код_товару = ?',
          [Кількість, Код_товару]
        );
      }

      await connection.commit();

      res.status(200).json({
        success: true,
        Код_замовлення: Код_замовлення,
        message: 'Замовлення успішно створено'
      });
    } catch (err) {
      await connection.rollback();
      throw err;
    } finally {
      connection.release();
    }
  } catch (err) {
    console.error('Помилка створення замовлення:', err);

    res.status(500).json({
      error: 'Помилка створення замовлення',
      details: err.message,
      sqlError: err.sqlMessage
    });
  }
});

module.exports = router;

const express = require('express');
const router = express.Router();
const db = require('../db');

router.get('/', async (req, res) => {
```

```
  try {
    const [rows] = await db.query('SELECT * FROM Товар');
    res.json(rows);
  } catch (err) {
    console.error('Помилка при завантаженні товарів:', err);
    res.status(500).json({ error: 'Помилка сервера' });
  }
});

router.get('/:id', async (req, res) => {
  try {
    console.log("Запит на товар з ID:", req.params.id);
    const [rows] = await db.query('SELECT * FROM Товар WHERE
Код_товару = ?', [req.params.id]);
    if (rows.length === 0) return res.status(404).json({ error:
'Товар не знайдено' });
    res.json(rows[0]);
  } catch (err) {
    console.error('Помилка при отриманні товару:', err);
    res.status(500).json({ error: 'Помилка сервера' });
  }
});

module.exports = router;


const express = require('express');
const cors = require('cors');
const app = express();
const pool = require('./db');

app.use(cors());
app.use(express.json());

app.get('/api/branches', async (req, res) => {
  const [rows] = await pool.query('SELECT * FROM Відділення');
  res.json(rows);
});


app.use('/api/clients', require('./routes/clients'));
app.use('/api/orders', require('./routes/orders'));
app.use('/api/products', require('./routes/products'));
```

```
app.use('/api/branches', require('./routes/branches'));
app.use('/api/details', require('./routes/details'));


const PORT = 5000;
app.listen(PORT, () => {
  console.log(`Server is running on http://localhost:${PORT}`);
});
```

Додаток Б

Код frontend-частини сайту

```
import { useState } from 'react';
import axios from 'axios';

export default function OrderStatus() {
  const [orderId, setOrderId] = useState('');
  const [order, setOrder] = useState(null);
  const [error, setError] = useState('');

  const handleCheckOrder = async (e) => {
    e.preventDefault();
    setError('');
    setOrder(null);

    try {
      const res = await
axios.get(`http://localhost:5000/api/details/${orderId}`);
      setOrder(res.data);
    } catch (err) {
      setError('Замовлення не знайдено або сталася помилка.');
    }
  };

  return (
    <div className="max-w-md mx-auto p-4">
      <form onSubmit={handleCheckOrder} className="space-y-4">
        <input
          type="number"
          placeholder="Введіть код замовлення"
          value={orderId}
          onChange={(e) => setOrderId(e.target.value)}
          required
          className="input"
        />
        <button type="submit" className="btn btn-primary w-full">
          Перевірити замовлення
        </button>
      </form>

      {order && (
        <div className="mt-6 card p-4">
```

Продовження Додатку Б

```
            <h2 className="text-lg font-semibold mb-2">Інформація
про замовлення</h2>
            <p><strong>Код                    замовлення:</strong>
{order.Код_замовлення}</p>
            <p><strong>Дата:</strong> {order.Дата_замовлення}</p>
            <p><strong>Статус:</strong>  <span  className="badge
bg-primary text-white">{order.Статус}</span></p>
            <p><strong>Сума:</strong> {order.Сума} грн</p>

            <h3 className="mt-4 font-semibold">Товари:</h3>
            <ul className="space-y-2 mt-2">
            {order.Товари.map((товар, index) => (
                <li key={index} className="card p-2">
                <p><strong>Назва:</strong> {товар.Назва}</p>
                <p><strong>Кількість:</strong>
{товар.Кількість}</p>
                <p><strong>Ціна:</strong> {товар.Ціна} грн</p>
                <p><strong>Вартість:</strong>    {товар.Вартість}
грн</p>
                </li>
            ))}
            </ul>
        </div>
    )}

      {error && <p className="mt-4 text-secondary">{error}</p>}
    </div>
  );
}

import categories from './data/categories';
import { useState } from 'react';

export default function Filters({ selected, onSelect }) {
  const [isOpen, setIsOpen] = useState(false);

  if (!categories || categories.length === 0) return null;

  const toggleFilters = () => {
    setIsOpen(!isOpen);
  };

  return (
    <aside className="w-full md:w-64">
```

Продовження Додатку Б

```
      <div className="card overflow-visible">
        <div className="p-4 bg-primary text-white flex justify-
between items-center md:hidden" onClick={toggleFilters}>
          <h2 className="font-bold text-lg">Фільтри</h2>
          <svg xmlns="http://www.w3.org/2000/svg" className={`h-5
w-5 transition-transform ${isOpen ? 'transform rotate-180' : ''}`}
fill="none" viewBox="0 0 24 24" stroke="currentColor">
            <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M19 9l-7 7-7-7" />
          </svg>
        </div>

        <div className={`p-4 md:block ${isOpen ? 'block' :
'hidden'}`}>
          <h2 className="font-bold mb-4 text-lg text-primary
hidden md:block">Категорії</h2>

          <div className="mb-4">
            <button
              className={`w-full text-left py-2 px-3 mb-2 rounded
${selected === null ? 'bg-primary text-white' : 'hover:bg-
light'}`}
              onClick={() => onSelect(null)}
            >
              Всі категорії
            </button>
          </div>

          <ul className="space-y-1">
            {categories.map((cat) => (
              <li key={cat.name}>
                <button
                  className={`w-full text-left py-2 px-3 rounded
transition ${
                    selected === cat.name ? 'bg-primary text-
white' : 'hover:bg-light'
                  }`}
                  onClick={() => onSelect(cat.name)}
                >
                  {cat.name}
                </button>
                {selected === cat.name &&
Array.isArray(cat.subcategories) && (
                  <ul className="ml-4 mt-2 space-y-1 text-sm">
```

```
                    {cat.subcategories.map((sub) => (
                      <li key={sub}>
                        <button
                          className={`w-full text-left py-1 px-3
rounded transition ${
                            selected === sub ? 'bg-light font-
medium text-primary' : 'hover:bg-light'
                          }`}
                          onClick={() => onSelect(sub)}
                        >
                          {sub}
                        </button>
                      </li>
                    ))}
                  </ul>
                )}
              </li>
            ))}
          </ul>
        </div>
      </div>
    </aside>
  );
}

export default function Footer() {
  return (
    <footer className="bg-primary text-white mt-12">
      <div className="container mx-auto px-4 py-8">
        <div className="grid grid-cols-1 md:grid-cols-3 gap-8">
          <div>
            <h3   className="text-lg   font-bold   mb-4">Онлайн
Аптека</h3>
            <p   className="text-light">Ваше   здоров'я   -   наш
пріоритет. Ми забезпечуємо швидку доставку якісних ліків по всій
Україні.</p>
          </div>
```

Продовження Додатку Б

```
          <div>
            <h3   className="text-lg   font-bold   mb-4">Швидкі
посилання</h3>
            <ul className="space-y-2">
              <li><a    href="/catalog"    className="text-light
hover:text-white transition">Каталог</a></li>
              <li><a  href="/order-status"  className="text-light
hover:text-white transition">Перевірити замовлення</a></li>
              <li><a     href="/cart"     className="text-light
hover:text-white transition">Кошик</a></li>
            </ul>
          </div>

          <div>
            <h3 className="text-lg font-bold mb-4">Контакти</h3>
            <ul className="space-y-2 text-light">
              <li className="flex items-center">
                <svg          xmlns="http://www.w3.org/2000/svg"
className="h-5  w-5  mr-2"  fill="none"  viewBox="0  0  24  24"
stroke="currentColor">
                  <path                     strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M3 5a2 2 0 012-2h3.28a1
1 0 01.948.684l1.498 4.493a1 1 0 01-.502 1.21l-2.257 1.13a11.042
11.042 0 005.516 5.516l1.13-2.257a1 1 0 011.21-.502l4.493 1.498a1
1 0 01.684.949V19a2 2 0 01-2 2h-1C9.716 21 3 14.284 3 6V5z" />
                </svg>
                +380 (50) 123-4567
              </li>
              <li className="flex items-center">
                <svg          xmlns="http://www.w3.org/2000/svg"
className="h-5  w-5  mr-2"  fill="none"  viewBox="0  0  24  24"
stroke="currentColor">
                  <path                     strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M3 8l7.89 5.26a2 2 0
002.22 0L21 8M5 19h14a2 2 0 002-2V7a2 2 0 00-2-2H5a2 2 0 00-2
2v10a2 2 0 002 2z" />
                </svg>
                info@pharma.ua
              </li>
              <li className="flex items-center">
```

Продовження Додатку Б

```
                <svg           xmlns="http://www.w3.org/2000/svg"
className="h-5  w-5  mr-2"  fill="none"  viewBox="0  0  24  24"
stroke="currentColor">
                  <path                      strokeLinecap="round"
strokeLinejoin="round"  strokeWidth={2}  d="M17.657  16.657L13.414
20.9a1.998 1.998 0 01-2.827 0l-4.244-4.243a8 8 0 1111.314 0z" />
                  <path                      strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M15 11a3 3 0 11-6 0 3 3
0 016 0z" />
                </svg>
                м. Київ, вул. Хрещатик, 1
              </li>
            </ul>
          </div>
        </div>

        <div className="mt-8 pt-6 border-t border-gray-700 text-
center text-sm text-light">
          &copy; 2025 Онлайн Аптека. Всі права захищено.
        </div>
      </div>
    </footer>
  );
}

import { Link, useNavigate } from 'react-router-dom';
import { useState } from 'react';
import { useCart } from '../context/CartContext';

export default function Header() {
  const [query, setQuery] = useState('');
  const navigate = useNavigate();
  const { cart } = useCart();

  const cartItemsCount = cart.reduce((total, item) => total +
item.quantity, 0);

  const handleSearch = (e) => {
    e.preventDefault();
    navigate(`/catalog?search=${query}`);
  };

  return (
    <header className="bg-primary text-white shadow-md">
```

Продовження Додатку Б

```
      <div className="container mx-auto px-4 py-3">
        <div className="flex flex-col md:flex-row justify-between
items-center gap-4">
          <div  className="flex  items-center  gap-6  w-full  md:w-
auto">
            <Link to="/" className="text-2xl font-bold flex items-
center">
              <svg             xmlns="http://www.w3.org/2000/svg"
className="h-8  w-8  mr-2"  fill="none"  viewBox="0  0  24  24"
stroke="currentColor">
                <path                        strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M19 21V5a2 2 0 00-2-
2H7a2 2 0 00-2 2v16m14 0h2m-2 0h-5m-9 0H3m2 0h5M9 7h1m-1 4h1m4-
4h1m-1 4h1m-5 10v-5a1 1 0 011-1h2a1 1 0 011 1v5m-4 0h4" />
              </svg>
              <span>Pharma</span>
            </Link>
            <nav className="hidden md:flex gap-6 text-sm sm:text-
base">
              <Link      to="/"      className="hover:text-light
transition">Головна</Link>
              <Link   to="/catalog"   className="hover:text-light
transition">Каталог</Link>
              <Link   to="/details"   className="hover:text-light
transition">Статус замовлення</Link>
            </nav>
          </div>

          <div  className="flex  items-center  gap-4  w-full  md:w-
auto">
            <form  onSubmit={handleSearch}  className="flex-grow
md:flex-grow-0 max-w-md">
              <div className="relative">
                <input
                  type="text"
                  placeholder="Пошук ліків..."
                  value={query}
                  onChange={(e) => setQuery(e.target.value)}
                  className="input text-primary w-full pr-10"
                />
                <button
                  type="submit"
                  className="absolute  right-0  top-0  h-full  px-3
text-gray-500 hover:text-primary"
                >
```

Продовження Додатку Б

```
                <svg         xmlns="http://www.w3.org/2000/svg"
className="h-5   w-5"   fill="none"   viewBox="0   0   24   24"
stroke="currentColor">
                  <path                   strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M21 21l-6-6m2-5a7 7 0
11-14 0 7 7 0 0114 0z" />
                </svg>
              </button>
            </div>
          </form>

          <Link to="/cart" className="relative">
            <svg             xmlns="http://www.w3.org/2000/svg"
className="h-6   w-6"   fill="none"   viewBox="0   0   24   24"
stroke="currentColor">
              <path                         strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M3 3h2l.4 2M7 13h10l4-
8H5.4M7   13L5.4   5M7   13l-2.293   2.293c-.63.63-.184   1.707.707
1.707H17m0 0a2 2 0 100 4 2 2 0 000-4zm-8 2a2 2 0 11-4 0 2 2 0 014
0z" />
            </svg>
            {cartItemsCount > 0 && (
              <span  className="absolute  -top-2  -right-2  bg-
secondary text-white text-xs font-bold rounded-full h-5 w-5 flex
items-center justify-center">
                {cartItemsCount}
              </span>
            )}
          </Link>

          <button    className="md:hidden"    onClick={()    =>
document.getElementById('mobile-
menu').classList.toggle('hidden')}>
            <svg             xmlns="http://www.w3.org/2000/svg"
className="h-6   w-6"   fill="none"   viewBox="0   0   24   24"
stroke="currentColor">
              <path                         strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M4 6h16M4 12h16M4 18h16"
/>
            </svg>
          </button>
        </div>
      </div>
```

Продовження Додатку Б

```
        <div id="mobile-menu" className="md:hidden hidden pt-4 pb-
2">
          <nav className="flex flex-col gap-3">
            <Link       to="/"       className="hover:text-light
transition">Головна</Link>
            <Link    to="/catalog"    className="hover:text-light
transition">Каталог</Link>
            <Link  to="/order-status"  className="hover:text-light
transition">Статус замовлення</Link>
          </nav>
        </div>
      </div>
    </header>
  );
}

import { Link } from 'react-router-dom';
import { useCart } from '../context/CartContext';

export default function ProductCard({ product }) {
  const { addToCart } = useCart();

  const handleAddToCart = (e) => {
    e.preventDefault();
    e.stopPropagation();
    addToCart({
      id: product.id,
      name: product.name,
      category: product.category,
      price: product.price,
      quantity: 1
    });
  };

  return (
    <div className="card h-full flex flex-col">
      <Link  to={`/product/${product.id}`}  className="block  h-
full">
        <div className="p-4 flex flex-col h-full">
          <div className="bg-light rounded-lg p-4 mb-4 flex items-
center justify-center">

            <svg xmlns="http://www.w3.org/2000/svg" className="h-
16  w-16  text-primary"  fill="none"  viewBox="0  0  24  24"
stroke="currentColor">
```

Продовження Додатку Б

```
              <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M19.428 15.428a2 2 0 00-1.022-.547l-2.387-
.477a6 6 0 00-3.86.517l-.318.158a6 6 0 01-3.86.517L6.05 15.21a2 2
0 00-1.806.547M8 4h8l-1 1v5.172a2 2 0 00.586 1.414l5 5c1.26
1.26.367 3.414-1.415 3.414H4.828c-1.782 0-2.674-2.154-1.414-
3.414l5-5A2 2 0 009 10.172V5L8 4z" />
            </svg>
          </div>

          <h3 className="font-bold text-lg text-primary mb-
1">{product.name}</h3>
          <span     className="text-sm     text-gray-600     mb-
2">{product.category}</span>
          <div className="flex items-baseline mt-auto">
            <span     className="text-lg     font-bold     text-
secondary">{product.price} ₴</span>
          </div>
        </div>

        <div className="mt-auto border-t p-3">
          <button
            onClick={handleAddToCart}
            className="w-full bg-primary hover:bg-opacity-90
text-white py-2 rounded-md flex items-center justify-center
transition"
          >
            <svg xmlns="http://www.w3.org/2000/svg" className="h-
5 w-5 mr-1" fill="none" viewBox="0 0 24 24" stroke="currentColor">
              <path strokeLinecap="round" strokeLinejoin="round"
strokeWidth={2} d="M3 3h2l.4 2M7 13h10l4-8H5.4M7 13L5.4 5M7 13l-
2.293 2.293c-.63.63-.184 1.707.707 1.707H17m0 0a2 2 0 100 4 2 2 0
000-4zm-8 2a2 2 0 11-4 0 2 2 0 014 0z" />
            </svg>
            Додати в кошик
          </button>
        </div>
      </Link>
    </div>
  );
}
```

```
import Filters from '../components/Filters';
import ProductCard from '../components/ProductCard';
import { useState, useEffect } from 'react';
import axios from 'axios';
import { useLocation } from 'react-router-dom';

export default function Catalog() {
  const [selected, setSelected] = useState(null);
  const [products, setProducts] = useState([]);
  const [loading, setLoading] = useState(true);
  const location = useLocation();

  const searchQuery = new
URLSearchParams(location.search).get('search') || '';

  useEffect(() => {
    setLoading(true);
    axios.get('http://localhost:5000/api/products')
      .then(res => {
        setProducts(res.data);
        setLoading(false);
      })
      .catch(err => {
        console.error('Помилка завантаження товарів:', err);
        setLoading(false);
      });
  }, []);

  const filteredProducts = products
    .filter(p => !selected || p.Категорія === selected)
    .filter(p => !searchQuery ||
p.Назва.toLowerCase().includes(searchQuery.toLowerCase()));

  return (
    <div className="container mx-auto px-4 py-6">
      <h1 className="text-2xl md:text-3xl font-bold text-primary
mb-6">Каталог товарів</h1>

      {searchQuery && (
        <div className="mb-4 p-3 bg-light rounded">
          <p className="text-gray-600">

            Результати пошуку для: <span className="font-
semibold">{searchQuery}</span>
```

```
          </p>
        </div>
      )}

      <div className="flex flex-col md:flex-row gap-6">
        <Filters selected={selected} onSelect={setSelected} />

        <div className="flex-1">
          {loading ? (
            <div  className="flex  justify-center  items-center  h-
64">
              <svg  className="animate-spin  h-10  w-10  text-
primary"      xmlns="http://www.w3.org/2000/svg"      fill="none"
viewBox="0 0 24 24">
                <circle  className="opacity-25"  cx="12"  cy="12"
r="10" stroke="currentColor" strokeWidth="4"></circle>
                <path  className="opacity-75" fill="currentColor"
d="M4 12a8 8 0 018-8V0C5.373 0 0 5.373 0 12h4zm2 5.291A7.962 7.962
0 014 12H0c0 3.042 1.135 5.824 3 7.938l3-2.647z"></path>
              </svg>
            </div>
          ) : filteredProducts.length === 0 ? (
            <div className="text-center py-10">
              <svg             xmlns="http://www.w3.org/2000/svg"
className="h-16  w-16  text-gray-400  mx-auto  mb-4"  fill="none"
viewBox="0 0 24 24" stroke="currentColor">
                <path                          strokeLinecap="round"
strokeLinejoin="round"  strokeWidth={2}  d="M9.172  16.172a4  4  0
015.656 0M9 10h.01M15 10h.01M21 12a9 9 0 11-18 0 9 9 0 0118 0z"
/>
              </svg>
              <h3   className="text-xl   font-medium   text-gray-
500">Товарів не знайдено</h3>
              <p className="text-gray-500 mt-1">Спробуйте змінити
параметри пошуку</p>
            </div>
          ) : (
            <div   className="grid   grid-cols-1   sm:grid-cols-2
lg:grid-cols-3 gap-6">
              {filteredProducts.map(product => (
                <ProductCard key={product.Код_товару} product={{
                  id: product.Код_товару,
                  name: product.Назва,
                  category: product.Категорія,
                  price: product.Ціна
```

Продовження Додатку Б

```
                }} />
              ))}
            </div>
          )}
        </div>
      </div>
    </div>
  );
}

import { Link } from 'react-router-dom';
export default function Home() {
  return (
    <div className="space-y-12">
      <section className="bg-primary text-white py-12 md:py-20">
        <div className="container mx-auto px-4">
          <div  className="flex  flex-col  md:flex-row  items-
center">
            <div className="md:w-1/2 mb-8 md:mb-0">
              <h1  className="text-3xl  md:text-4xl  lg:text-5xl
font-bold mb-4">
                Ласкаво просимо до Онлайн Аптеки
              </h1>
              <p className="text-lg md:text-xl text-light mb-6">
                Швидке  замовлення  ліків  з  доставкою  або
самовивозом. Ваше здоров'я - наш пріоритет.
              </p>
              <Link  to="/catalog"  className="btn  btn-secondary
inline-block">
                Перейти в каталог
              </Link>
            </div>
            <div className="md:w-1/2">
          <img
src="https://buzzrx.s3.amazonaws.com/a0215ed5-6bf5-405f-b6ab-
b43667abacad/2017-10-31_OnlineMeds.png"
                alt="Онлайн аптека"
                className="rounded-lg shadow-xl mx-auto"
              />
            </div>
          </div>
        </div>
      </section>

      <section className="container mx-auto px-4">
```

Продовження Додатку Б

```
        <h2 className="text-2xl md:text-3xl font-bold text-
primary text-center mb-8">
          Чому обирають нас
        </h2>

        <div className="grid grid-cols-1 md:grid-cols-3 gap-8">
          <div className="card p-6 text-center">
            <div className="bg-primary bg-opacity-10 p-4 rounded-
full w-16 h-16 flex items-center justify-center mx-auto mb-4">
              <svg xmlns="http://www.w3.org/2000/svg"
className="h-8 w-8 text-primary" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                <path strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M12 8v4l3 3m6-3a9 9 0
11-18 0 9 9 0 0118 0z" />
              </svg>
            </div>
            <h3 className="text-xl font-bold text-primary mb-
2">Швидка доставка</h3>
            <p     className="text-gray-600">Доставляємо     ліки
протягом 24 годин після оформлення замовлення.</p>
          </div>

          <div className="card p-6 text-center">
            <div className="bg-primary bg-opacity-10 p-4 rounded-
full w-16 h-16 flex items-center justify-center mx-auto mb-4">
              <svg xmlns="http://www.w3.org/2000/svg"
className="h-8 w-8 text-primary" fill="none" viewBox="0 0 24 24"
stroke="currentColor">

                <path strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M9 12l2 2 4-4m5.618-
4.016A11.955 11.955 0 0112 2.944a11.955 11.955 0 01-8.618
3.04A12.02 12.02 0 003 9c0 5.591 3.824 10.29 9 11.622 5.176-1.332
9-6.03 9-11.622 0-1.042-.133-2.052-.382-3.016z" />
              </svg>
            </div>
            <h3 className="text-xl font-bold text-primary mb-
2">Гарантія якості</h3>
            <p      className="text-gray-600">Всі      препарати
сертифіковані та зберігаються в правильних умовах.</p>
```

Продовження Додатку Б

```
            <div className="bg-primary bg-opacity-10 p-4 rounded-
full w-16 h-16 flex items-center justify-center mx-auto mb-4">
              <svg             xmlns="http://www.w3.org/2000/svg"
className="h-8 w-8 text-primary" fill="none" viewBox="0 0 24 24"
stroke="currentColor">
                <path                          strokeLinecap="round"
strokeLinejoin="round" strokeWidth={2} d="M3 10h18M7 15h1m4 0h1m-
7 4h12a3 3 0 003-3V8a3 3 0 00-3-3H6a3 3 0 00-3 3v8a3 3 0 003 3z"
/>
              </svg>
            </div>
            <h3  className="text-xl  font-bold  text-primary  mb-
2">Зручна оплата</h3>
            <p  className="text-gray-600">Різні  способи  оплати,
включаючи оплату при отриманні.</p>
          </div>
        </div>
      </section>

      <section className="bg-light py-10">
        <div className="container mx-auto px-4 text-center">
          <h2  className="text-2xl  md:text-3xl  font-bold  text-
primary mb-4">
            Не гайте час - оформіть замовлення онлайн
          </h2>
          <p className="text-lg text-gray-600 mb-6 max-w-2xl mx-
auto">

            Наша онлайн-аптека пропонує широкий асортимент ліків
за доступними цінами з доставкою до вашого відділення.
          </p>
          <Link to="/catalog" className="btn btn-primary inline-
block">
            Перейти в каталог
          </Link>
        </div>
      </section>
    </div>
  );
}
```