

Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут комп'ютерних наук та інформаційних технологій
Кафедра автоматизованих систем управління



КУРСОВА РОБОТА

з навчальної дисципліни

“ Проектування інформаційних систем ”

для студентів спеціальності 122 «Комп'ютерні науки» (Обчислювальний
інтелект смарт-систем)

на тему “ Система автоматизації роботи аптеки ”

Студент гр. ОІ-36 Іванков Н.В.

Керівник: Ковалів Т.А.

Курсова захищена з оцінкою

“ _____ ”

“ ____ ” _____ 20__ р.

Львів – 2025

Зміст

Вступ.....	3
Технічне завдання.....	4
1) Теоретичний аналіз особливостей предметної області дослідження.....	6
1.1) Аналіз інформаційних потреб та визначення предметної області дослідження.....	6
1.2) Обґрунтування вибору засобів реалізації.....	7
2) Проєктування та реалізація веб-орієнтованої системи за обраною тематикою курсової роботи.....	9
2.1) Проєктування структури бази даних інформаційної системи за напрямком курсової роботи.....	9
2.2) Розробка алгоритмів обробки даних в інформаційній системі (бек-енд, фронт-енд, API тощо).....	12
2.3) Проєктування веб-інтерфейсу для роботи з даними.....	13
3) Інтерфейс та порядок роботи з веб-орієнтованою системою.....	15
3.1) Мінімальні технічні параметри, для розробленої системи.....	15
3.2) Порядок встановлення та налаштування системи.....	16
3.3) Структура інтерфейсу та порядок роботи із системою.....	17
3.4) Тестування роботи інформаційної системи.....	24
4) Концептуальне проєктування предметної області.....	25
5) Висновки.....	30
6) Список використаних джерел.....	31
7) Додатки.....	32
7.1) Додаток А.....	32
7.2) Додаток Б.....	37

Вступ

Актуальність теми курсової роботи зумовлена зростанням популярності електронної комерції, зокрема веб-орієнтованих систем інтернет-магазинів. Сучасні аптеки є магазинами першої необхідності, і створення зручних платформ для продажу ліків онлайн сприяє підвищенню ефективності торговельних процесів, доступності продукції для споживачів та автоматизації бізнес-процесів.

Метою курсової роботи є дослідження особливостей проєктування та реалізації веб-орієнтованої системи автоматизації роботи аптеки.

Завданнями курсової роботи є:

1. Аналіз теоретичних засад проєктування та реалізації веб-орієнтованих систем.
2. Визначення інформаційних потреб предметної області.
3. Оцінка ризиків інформаційних потоків та аналіз їх структури.
4. Реалізація алгоритмів обробки даних та інтерфейсу системи для зручності користувачів.

Об'єктом проєктування є процес автоматизації роботи аптеки.

Предметом проєктування є методи та засоби, що забезпечують автоматизацію бізнес-процесів, пов'язаних з продажем і управлінням товаром.

ТЕХНІЧНЕ ЗАВДАННЯ

на курсовий проект

з дисципліни “ Проектування інформаційних систем ”

Прізвище, ім’я студента: Іванков Нікіта
Група: ОІ-36
Тема курсової роботи: *Система автоматизації роботи аптеки*

В рамках курсової роботи передбачається виконання таких завдань:

- Визначення предметної області: Визначення вимог до функціональності системи для автоматизації роботи аптеки, включаючи методи сортування та пошуку товарів, їх відображення на окремій сторінці та оформлення замовлення.
- Аналіз структури даних: Розробка структури бази даних для зберігання інформації про товари та замовлення.
- Розробка бази даних: Проектування та створення нереляційної бази даних з використанням MongoDB для зберігання необхідних даних.
- Реалізація серверної частини: Розробка скриптів для взаємодії з базою даних та обробки запитів користувачів.
- Реалізація клієнтської частини: Використання HTML, CSS, React та мови програмування JavaScript для розробки веб-ресурсу, що дає змогу переглядати, сортувати та купляти необхідні ліки.

- Інтеграція фронкенду та бекенду: Забезпечення взаємодії між клієнтською та серверною частинами для обміну даними.
- Аналіз сучасних технологій: Аналіз сучасних технологій веб-розробки, які можуть бути застосовані для створення системи автоматизації роботи аптеки.
- Середовище функціонування проєкту – Visual Studio Code

Термін здачі студентом роботи: _____.

Дата видачі завдання: ____.

Керівник роботи: Ковалів Т.А.

Завдання прийняв до виконання студент: Іванков Н.

1. Теоретичний аналіз особливостей предметної області дослідження

1.1 Аналіз інформаційних потреб та визначення предметної області дослідження

Предметною областю дослідження є автоматизація роботи аптеки для продажу ліків – товарів першої необхідності. Ця сфера має дуже важливе значення, оскільки відповідає за здоров'я людей.

Основні **вимоги** до інформаційного забезпечення цієї предметної області включають:

- Точність та актуальність даних: інформація про ліки має бути правильною та актуальною.
- Легкість доступу до інформації: користувачі повинні мати змогу легко шукати необхідні ліки за категоріями або через пошук.

Напрямки використання технологій для створення веб-орієнтованих інформаційних систем для аптек використовуються наступні технології та інструменти:

- *HTML, CSS, JavaScript*: для створення інтерактивного та доступного веб-інтерфейсу.
- *React.js*: для організації переходів між сторінками, а також забезпечення зручного відображення даних.
- *Node.js*: як серверна платформа для обробки запитів користувачів та виконання необхідних операцій з даними.
- *MongoDD*: база даних яка забезпечує зберігання усієї інформації про ліки та замовлення.

Задача курсової роботи полягає в створенні веб-орієнтованої системи автоматизації роботи аптеки. **Основними етапами** її розв'язання є:

1. *Аналіз вимог предметної області*: оцінка потреб користувачів та визначення вимог до інформаційної системи.
2. *Проектування бази даних*: збереження даних про ліки
3. *Розробка інтерфейсу користувача*: створення зручного та доступного інтерфейсу для перегляду, фільтрації та купівлі ліків.
4. *Тестування та оптимізація*: перевірка роботи системи на всіх етапах її функціонування, забезпечення коректності роботи та ефективності.

Технічне завдання:

- *Перегляд інформації*: відображення основної інформації (назва, фото, ціна) про усі ліки.
- *Фільтрація*: можливість фільтрації ліків за категоріями
- *Пошук*: можливість шукати ліки на сайті за назвою
- *Перегляд повної інформації*: детальна сторінка кожного товару з повним описом та характеристиками.
- *Додавання та видалення з кошику*: можливість додавати кожен товар у кошик як з головної сторінки сайту, так і з сторінки товару. У кошику повинна відображатись кількість кожного товару, загальна вартість замовлення, можливість видалення товару з кошику та кнопка оформлення замовлення.
- *Оформлення замовлення*: можливість оформити замовлення із надсиланням листа із підтвердженням на пошту.

1.2 Обґрунтування вибору засобів реалізації

Для розробки веб-орієнтованої системи автоматизації роботи аптеки було здійснено порівняльний аналіз існуючих аналогічних систем. Метою аналізу було визначення оптимальних засобів реалізації, які найбільшою

мірою відповідають вимогам до функціональності, зручності використання, масштабованості та продуктивності.

Аналіз було проведено на прикладі таких систем, як:

- АНЦ (Аптека низьких цін):

Переваги: зручна структура даних, мультимедійна інтеграція

Недоліки: зовелика кількість контенту, погана оптимізація

- Аптека 9-1-1:

Переваги: зручна структура даних, зручний пошук

Недоліки: відсутність фільтрації, незручний інтерфейс

На основі аналізу аналогів і враховуючи вимоги до проєкту, було обрано такі інструменти та технології:

- *React i React Router:*

React забезпечує структурований спосіб представлення даних, інтерактивність та динамічний контент. React Router відповідає за маршрутизацію та зв'язок між сторінками.

- *CSS:*

CSS дозволяє реалізувати стильний і адаптивний дизайн для різних пристроїв.

- *Node.js:*

Забезпечує ефективну серверну частину для взаємодії з клієнтським інтерфейсом. Швидкодія та зручність у розробці REST API для інтеграції з фронтом.

- ***MongoDB:***

Дозволяє реалізувати нереляційну базу даних для зберігання інформації про товари та замовлення.

Обрані засоби реалізації, забезпечують оптимальний баланс між функціональністю, продуктивністю та простотою розробки.

2. Проєктування та реалізація веб-орієнтованої системи за обраною тематикою курсової роботи

2.1 Проєктування структури бази даних інформаційної системи за напрямком курсової роботи

Розробка бази даних є одним із ключових етапів проєктування. Структура бази даних повинна забезпечувати зручне зберігання, доступ та управління інформацією, а також задовольняти всі функціональні вимоги системи.

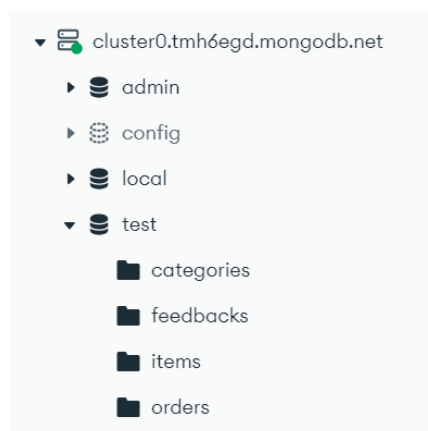


Рис 1. Загальний вигляд створеної БД

- **categories:** зберігання даних про категорії
- **feedbacks:** зберігання відгуків
- **items:** зберігання даних про товари
- **orders:** зберігання даних про замовлення

Структура таблиці **categories:**

Поле	Тип	Опис
id	ObjectID	Унікальний ідентифікатор категорії.
key	VARCHAR(255)	Ключ категорії.
name	VARCHAR(255)	Назва категорії

Структура таблиці **feedbacks:**

Поле	Тип	Опис
id	ObjectID	Унікальний ідентифікатор відгука.
email	VARCHAR(255)	Електронна пошта користувача
message	TEXT	Повний відгук
date	DATETIME	Дата та час відправлення повідомлення

Структура таблиці **items:**

Поле	Тип	Опис
id	ObjectID	Унікальний ідентифікатор товару.
title	VARCHAR(255)	Назва товару
img	TEXT	Base64-закодоване зображення
desc	TEXT	Опис товару
category	VARCHAR(255)	Категорія
price	DECIMAL(10,2)	Ціна
char1	VARCHAR(255)	Додаткова характеристика
char2	VARCHAR(255)	Додаткова характеристика
char3	VARCHAR(255)	Додаткова характеристика
char4	VARCHAR(255)	Додаткова характеристика

Зв'язок з таблицею categories – кожний товар має прив'язку до категорії через category.

Структура таблиці **orders**:

Поле	Тип	Опис
id	ObjectID	Унікальний ідентифікатор замовлення.

firstName	VARCHAR(255)	Ім'я користувача
lastName	VARCHAR(255)	Прізвище користувача
email	VARCHAR(255)	Електронна пошта користувача
phone	VARCHAR(20)	Телефон
items	ARRAY	Список товарів
total	DECIMAL (10,2)	Загальна сума замовлення
date	DATETIME	Час замовлення

2.2 Розробка алгоритмів обробки даних в інформаційній системі (бек-енд, фронт-енд, API тощо)

Основні процеси обробки даних включають:

1) *Отримання даних від користувача (сортування, пошук, додавання/видалення з козили, оформлення замовлення та написання відгуку):*

Користувач вводить запит (наприклад обирає категорію за якою хоче посортувати товар, шукає товар через пошук, додає чи видаляє товар з козили оформляє замовлення або пише відгук) на фронтенді. Цей запит передається через API до бекенду.

2) *Обробка запиту на сервері:*

Бекенд отримує запит, перевіряє його коректність (наявність даних,

формат) та формує запит до сховища даних. Запит може включати:

- Пошук за назвою: використовується пошук за унікальним ідентифікатором.
- Пошук за категорією: вибір товарів, що відповідають певній категорії.
- Підтвердження замовлення: підтвердження створеного клієнтом замовлення

3) Повернення даних до користувача:

Сервер формує JSON-відповідь із результатами пошуку або відправляє лист на пошту із підтвердженням замовлення. Фронтенд відображає отримані дані у вигляді списку, детальної сторінки про товар або елемента у корзині.

Загальний алгоритм роботи:

1. Користувач робить запит на фронтенді.
2. Дані запиту передаються на сервер через API.
3. Сервер обробляє запит і робить відповідний запит до бази даних.
4. Результати вибірки передаються у форматі JSON назад на фронтенд.
5. Фронтенд відображає результати на сторінці

2.3 Проєктування веб-інтерфейсу для роботи з даними

Структура інтерфейсу системи

Інтерфейс системи розроблено таким чином, щоб забезпечити зручність та зрозумілість роботи користувачів із товарами на сайті. Він складається з наступних компонентів:

- *Головна сторінка*: сторінка містить елементи навігації у шапці, постер для зацікавлення користувачів та каталог товарів із можливістю фільтрації. Є можливість переходу на сторінку детальної інформації кожного товару, сторінку “Про Нас” або зворотнього зв’язку, а також відкриття і закриття корзини.
- *Сторінка детальної інформації*: містить детальну інформацію про кожен товар, його назву, фото, ціну, опис товару, та додаткові характеристики. Є можливість повернення на головну сторінку, перехід на сторінки “Про Нас” або зворотнього зв’язку, а також відкриття і закриття корзини.
- *Сторінка ‘Про Нас’*: містить усю необхідну інформацію про аптеку, таку як графік роботи, як виконується доставка та оплата. Є можливість повернення на головну сторінку, перехід на сторінку зворотнього зв’язку, а також відкриття і закриття корзини.
- *Сторінка зворотнього зв’язку*: Містить форму для зворотнього зв’язку. Є можливість повернення на головну сторінку, перехід на сторінку “Про Нас”, а також відкриття і закриття корзини.

Засоби для реалізації функцій системи:

Для розробки інтерфейсу системи було використано такі технології:

- *React/React Router* – для створення структури сторінок та маршрутизації.
- *CSS* – для стилізації інтерфейсу
- *JavaScript* – для реалізації інтерактивності (фільтрація, додавання/видалення у корзині, обробка кліків)

Звітні форми системи:

Інформація у системі відображається у вигляді форм:

- *Список товарів:* містить назву, ціну і фото товару
- *Детальна інформація про товар:* повна інформація про обраний товар із можливістю повернення до списку.

Зразки цих форм наведено у додатках.

3. Інтерфейс та порядок роботи з веб-орієнтованою системою

3.1 Мінімальні технічні параметри, для розробленої системи

Для коректного функціонування системи необхідно забезпечити наведені нижче системні та апаратні характеристики:

Системні вимоги:

- Операційна система: Windows 10/11, macOS 10.15+, або будь-який дистрибутив Linux
- Процесор: 2 ядра, 1.8 ГГц або вище
- Оперативна пам'ять: 4 ГБ (рекомендовано 8 ГБ)
- Браузер: Google Chrome, Mozilla Firefox, або будь-який інший браузер із підтримкою сучасних веб-стандартів.
- Мережа: стабільне підключення до інтернету для роботи із сервером.

Програмні вимоги:

- Node.js (версія 20.11 або новіша)
- React.js (версія 19.0 або новіша)
- Менеджер пакетів npm (встановлюється разом із Node.js)
- Редактор коду (рекомендовано Visual Studio Code)

- База даних MongoDB (локальна установка або доступ до MongoDB Atlas)
- Пакет Mongoose
- Налаштування IP-білосписку (whitelist) для підключення до MongoDB Atlas (якщо використовується хмарний кластер)

3.2 Порядок встановлення та налаштування системи

1. Завантажити архів з вихідним кодом проєкту.
2. Розархівувати проєкт
3. Встановити Node.js та React.js, якщо не встановлено (доступно на офіційних сайтах).
4. Запустити термінал\консоль, перейти до папки проєкту та виконати команду `npm install` для встановлення всіх залежностей.
5. Перейти у папку `server` і запустити серверну частину командою `node server.js`.
6. Перейти у папку `shop` і запустити клієнтську частину командою `npm start`.
7. Через MongoDB Atlas або MongoDB Compass підключитись до бази даних.
6. Відкрити браузер і перейти за адресою `http://localhost:3000`.

3.3 Структура інтерфейсу та порядок роботи із системою

Головна сторінка

Після запуску веб-сайту користувач потрапляє на головну сторінку. Тут можна переглянути каталог усіх товарів, перейти на сторінки “Про Нас” або зворотнього зв’язку, відкрити та закрити корзину.



Рис 1. Головна сторінка

Нижче на головній сторінці відображено каталог з усіма ліками, доступними для замовлення та списком категорій по яким можна сортувати товари. У каталозі відображено назву товару, його фото, ціну та кнопку додавання товару у корзину.

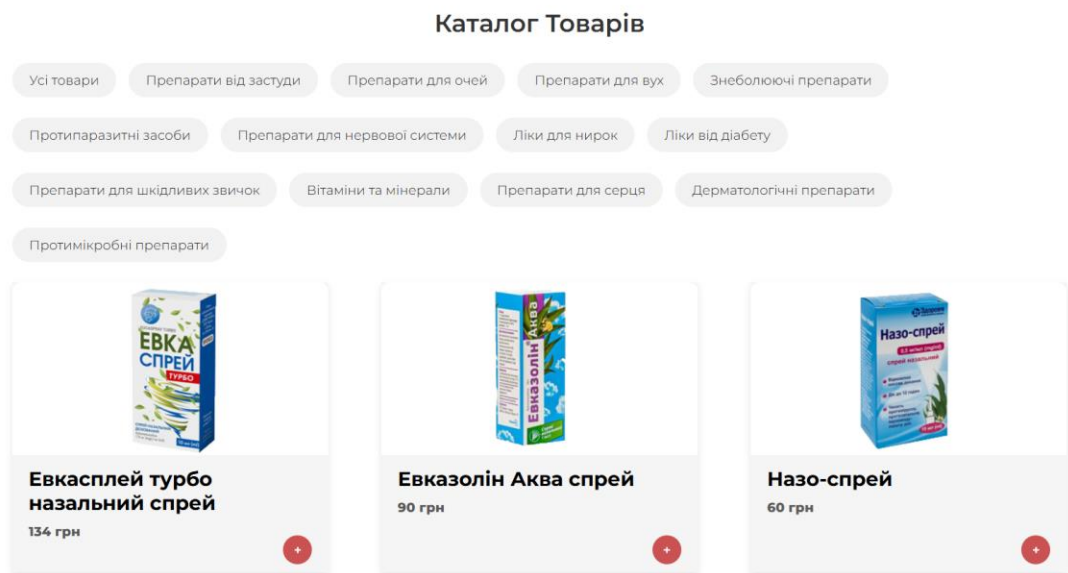


Рис 2. Каталог товарів

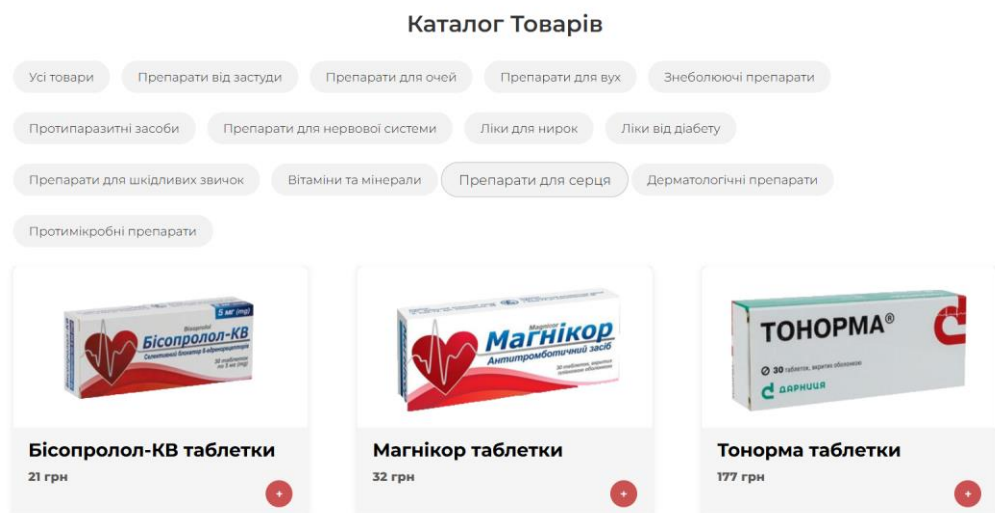


Рис 3. Каталог товарів при фільтрації за категорією

Натиснувши на червоний кружечок з + можна додати товар у корзину. Також при додаванні зверху з'явиться відповідне повідомлення. При додаванні одного товару декілька раз, його кількість у корзині буде збільшуватись. При натисканні на іконку смітника у корзині можна видалити товар з корзини.

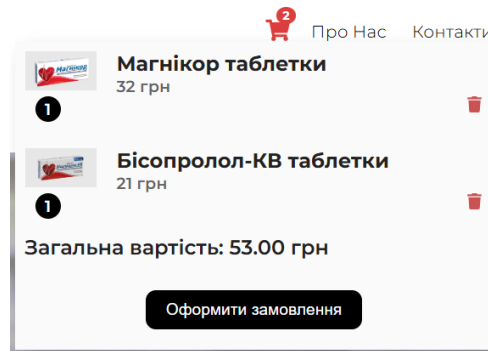


Рис 4. Додавання товарів у корзину

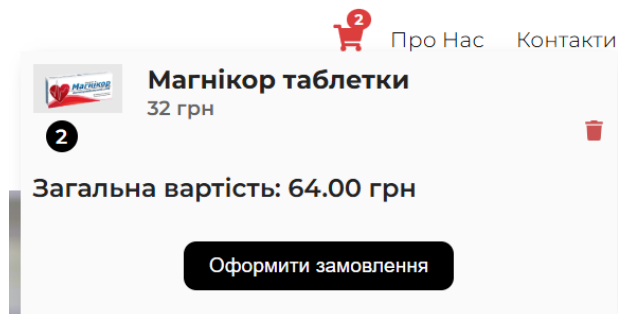


Рис 5. Додавання одного товару у корзину декілька разів

"Магнікор таблетки" додано у кошик!

Рис 6. Повідомлення про додавання товару у кошик

Також в кінці кожної сторінки присутній футер із навігацією та додатковою інформацією.

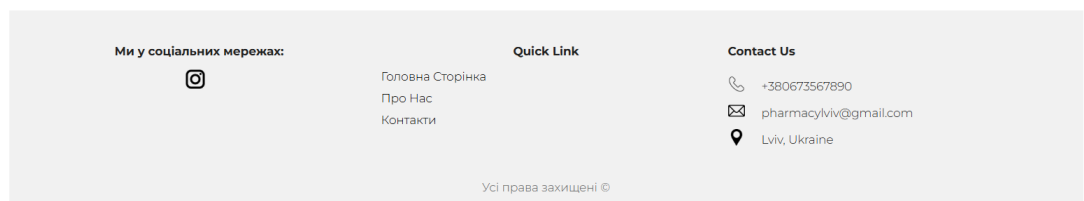


Рис 7. Футер

Сторінка оформлення замовлення

При натисканні на кнопку “Оформити замовлення” в корзині, користувача перекидає на окрему сторінку з підтвердженням замовлення. На сторінці є форма для введення даних, а також список усіх товарів у замовленні. При натисканні підтвердити замовлення на введену пошту буде відправлятися відповідний лист.





Оформлення замовлення	Ваше замовлення
Ім'я: <input type="text" value="Нікіта"/>	 Магнікор таблетки
Прізвище: <input type="text" value="Іванков"/>	Кількість: 2 Ціна за шт: 32 грн
Email: <input type="text" value="nikitavankov05@gmail.com"/>	Загальна вартість: 64.00 грн
Номер телефону: <input type="text" value="380673560623"/>	
<input type="button" value="Підтвердити замовлення"/>	

Рис 8. Сторінка оформлення замовлення



Підтвердження замовлення

pharmacylavka@gmail.com
кому мені ▼

Дякуємо за ваше замовлення, Нікіта!

Ми отримали ваше замовлення та вже почали формувати його.

Деталі замовлення:

- Магнікор таблетки (x2) - 32 грн

Загальна сума: 64.00 грн

З повагою, Аптечна лавка

Рис 9. Лист із підтвердженням замовлення

Сторінка “Про Нас”

Перейшовши на сторінку користувач зможе прочитати усю головну інформацію про аптеку, таку як графік роботи, умови доставки та оплати. Також на сторінці зверху є можливість повернутись на головну сторінку, або перейти на іншу сторінку, знизу є футер.



ABOUT US

Про Нас

Ми молода аптека у самому центрі Львова.
У нас на сайті Ви зможете знайти усі необхідні для Вас ліки по найкращим цінам.



Графік роботи

Пн-Пт: 08:00 - 23:00
Сб: 09:00 - 23:00
Нд: 09:00 - 21:00



Доставка та самовивіз

Доставка виконується кур'єром лише по Львову
Час доставки: 1-3 години з моменту замовлення

Самовивіз можливий із точки видачі при замовленні онлайн через 30 хв після створення замовлення



Оплата

Оплата можлива наступними варіантами:

- Оплата на сайті
- Оплата при отриманні у аптеці

Ми у соціальних мережах:



Головна Сторінка
Про Нас
Контакти

Quick Link

Contact Us



+380673567890
pharmacylviv@gmail.com
Lviv, Ukraine


Усі права захищені ©

Рис 10. Сторінка “Про нас”

Сторінка “Контакти”

Перейшовши на сторінку користувачу буде представлена форма в яку він зможе ввести свою електронну пошту, а також питання з яким він хоче звернутись до підтримки магазину.






[Про Нас](#)
[Контакти](#)

Зворотній зв'язок

Надіслати

Ми у соціальних мережах:



Quick Link

Головна Сторінка
Про Нас
Контакти

Contact Us



+380673567890
pharmacylviv@gmail.com
Lviv, Ukraine


Усі права захищені ©


Рис 11. Сторінка “Контакти”

Сторінка для перегляду детальної інформації про товар

На сторінці з детальною інформацією про товар, окрім назви, фото та ціни, відображається також детальний опис товару, а також є можливість додатково розгорнути список із характеристиками.


[Про Нас](#)
[Контакти](#)



Магнікор таблетки


32 грн

Магнікор таблетки вкриті плівковою оболонкою 75 мг білістер

Додати до кошика

[Відкрити характеристики](#)

Ми у соціальних мережах:



Quick Link

Головна Сторінка
Про Нас
Контакти

Contact Us

+380673567890
pharmacylviv@gmail.com
Lviv, Ukraine

Усі права захищені ©

Рис 12. Сторінка з детальною інформацією про товар

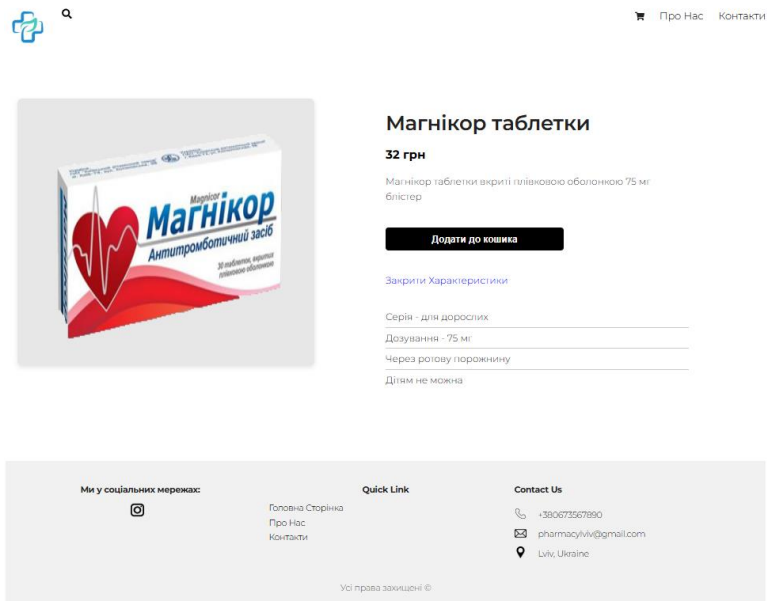


Рис 13. Сторінка з детальною інформацією про товар при розгорнутих характеристиках

Порядок роботи виглядає так:

1. Перегляд головної сторінки. Перехід на сторінку з інформацією або зворотнього зв'язку за потреби.
2. Перегляд каталогу товарів. Фільтрація або пошук товарів за необхідності
3. Додавання потрібних товарів у кошик
4. Перегляд кошику (Редагування за потреби)
5. Оформлення замовлення

3.4 Тестування роботи інформаційної системи

Процес тестування інформаційної системи охоплював як бекенд-компоненти, так і фронтенд-частину, з використанням актуальних методів і інструментів.

Основні підходи до тестування були наступними:

Тестування бекенду

Для перевірки бекенд-складової застосовувались автоматизовані юніт-тести:

- TestClient (FastAPI): використовується для моделювання HTTP-запитів до API.
- Тестування окремих функцій та API-ендпоінтів.
- Перевірка правильності реалізації бізнес-логіки.
- Тестування механізмів валідації даних для забезпечення коректності даних, введених користувачем.

Результати:

- Тестування показало стабільність роботи додатку при обробці стандартних запитів.

Тестування фронтенду

Технології: TypeScript, React. Фронтенд тестувався вручну з акцентом на функціональність користувацького інтерфейсу та взаємодію з API.

- Перевірка основних функцій інтерфейсу:
 - Фільтрація товарів
 - Пошук товарів
 - Додавання товарів до кошика та видалення їх з нього
 - Оформлення замовлення
- Перевірка коректності відображення даних:
 - Точність відображення карток товарів
 - Відповідність інформації у кошику введеним даним

- Правильне відображення даних на окремих сторінках товару
- Тестування взаємодії з бекендом:
 - Правильність надсилання запитів до API
 - Адекватна обробка помилок, таких як відсутність товару

Інструменти:

- Google Chrome для ручного тестування

Процес тестування:

- Тестування окремих компонентів на працездатність
- Аналіз взаємодії компонентів, наприклад, актуалізація кошика після додавання товару
- Оцінка зручності користувацького інтерфейсу (UI/UX)

Результати:

- Всі ключові функції працюють стабільно
- Усі виявлені візуальні недоліки під час тестування були успішно виправлені

4. Концептуальне проектування предметної області

IDEF0 (Функціональна декомпозиція):

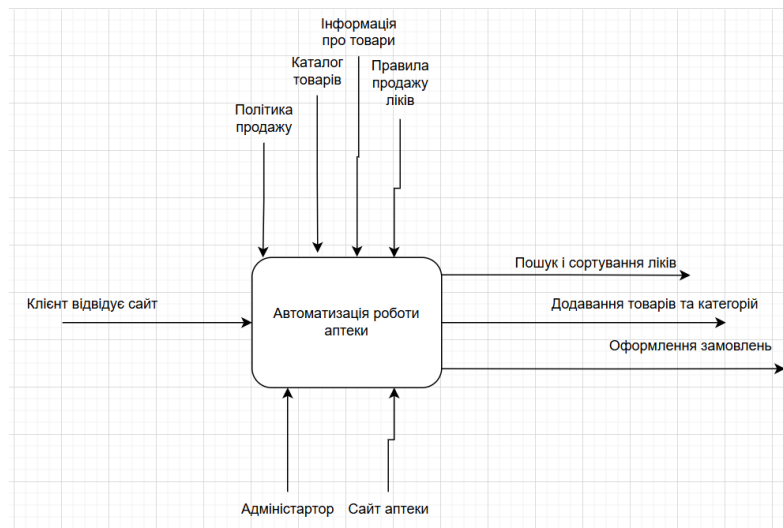


Рис. 14 - IDEF0 діаграма

DFD (Потоки даних):

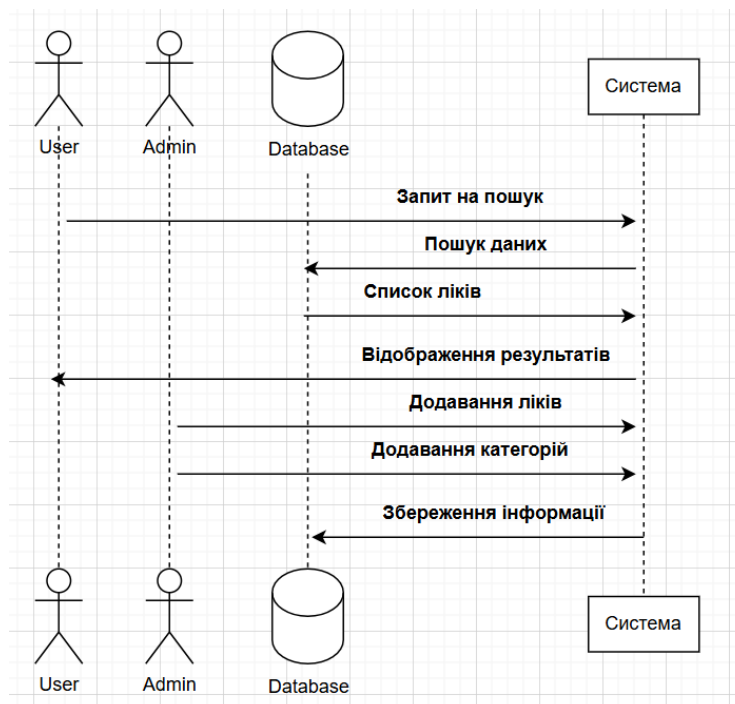


Рис. 15 – DFD діаграм

UML Use Case:

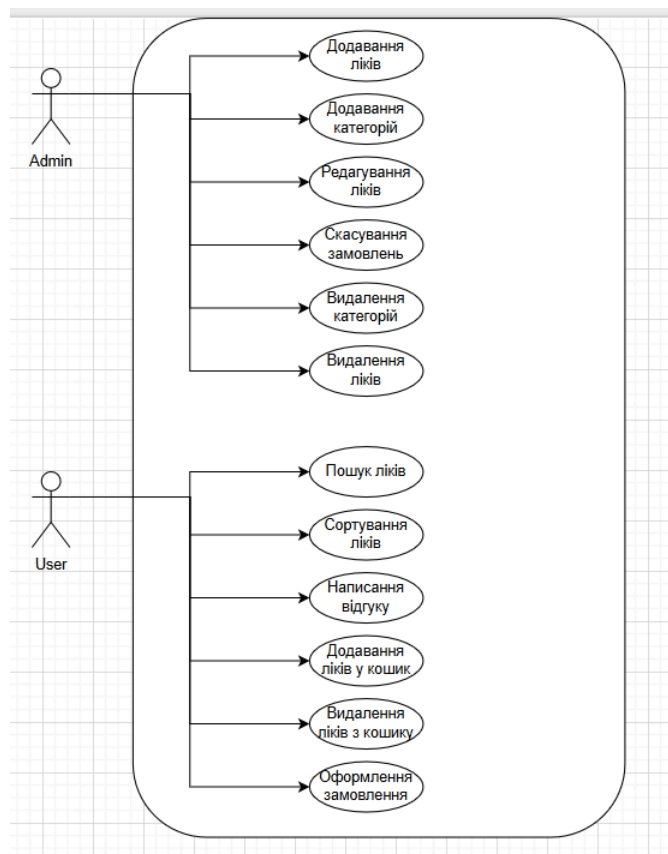


Рис. 16 - UML Use Case діаграма

IDEF1X (Структура бази даних):

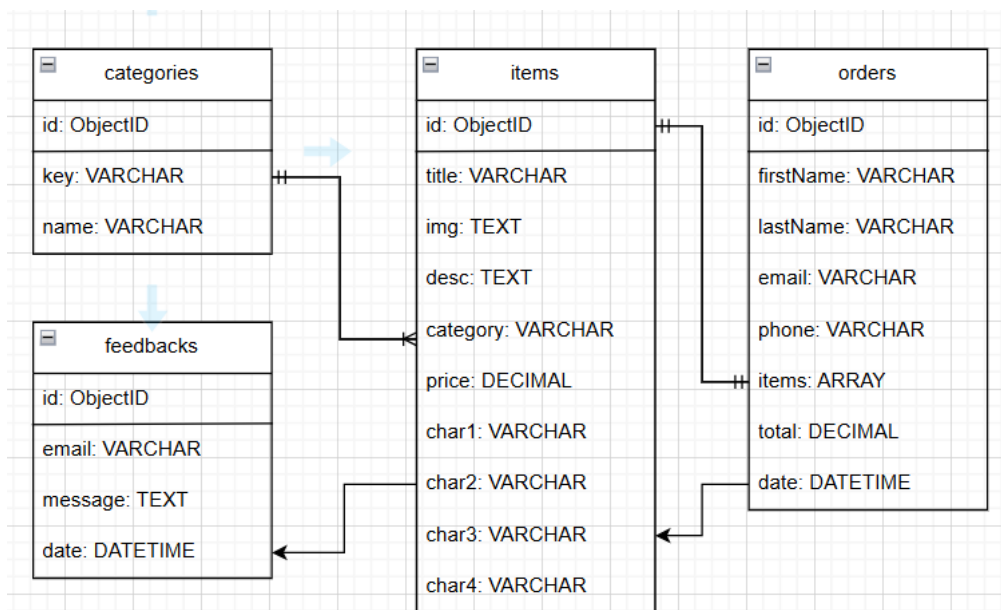


Рис. 17 IDEF1X діаграма

Діаграма класів:

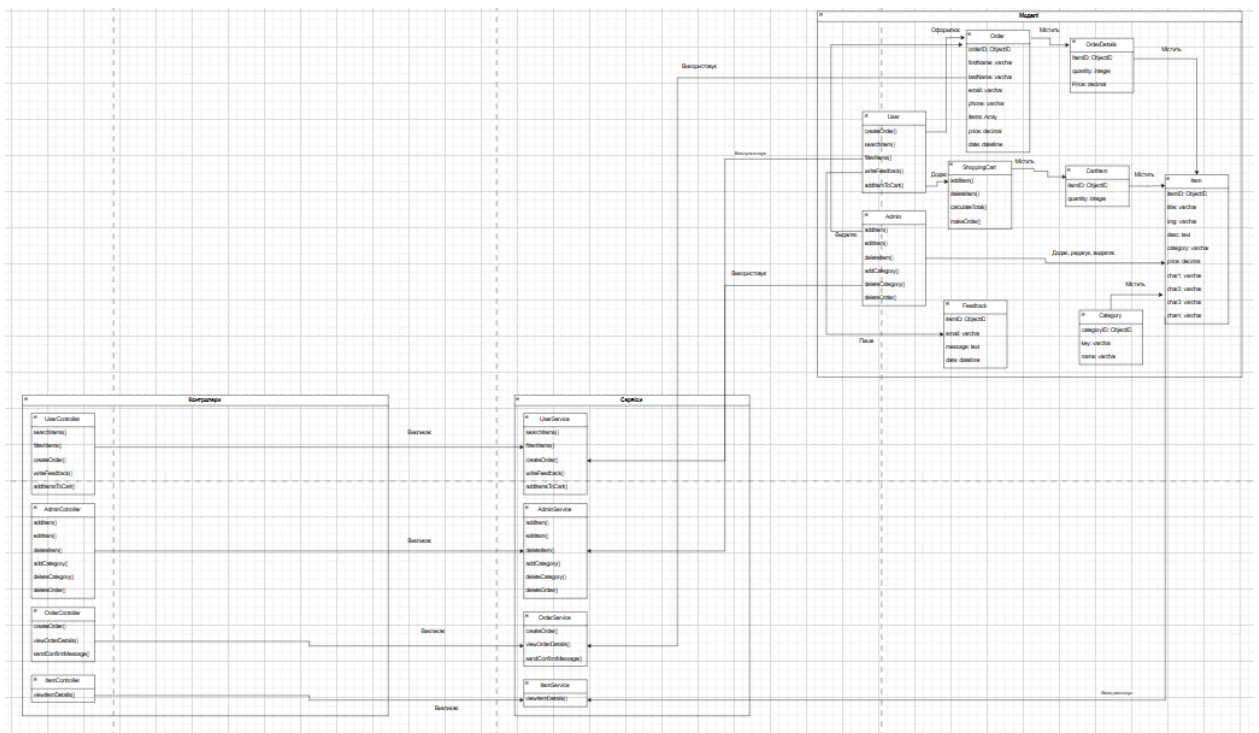


Рис. 18 IDEF1X діаграма

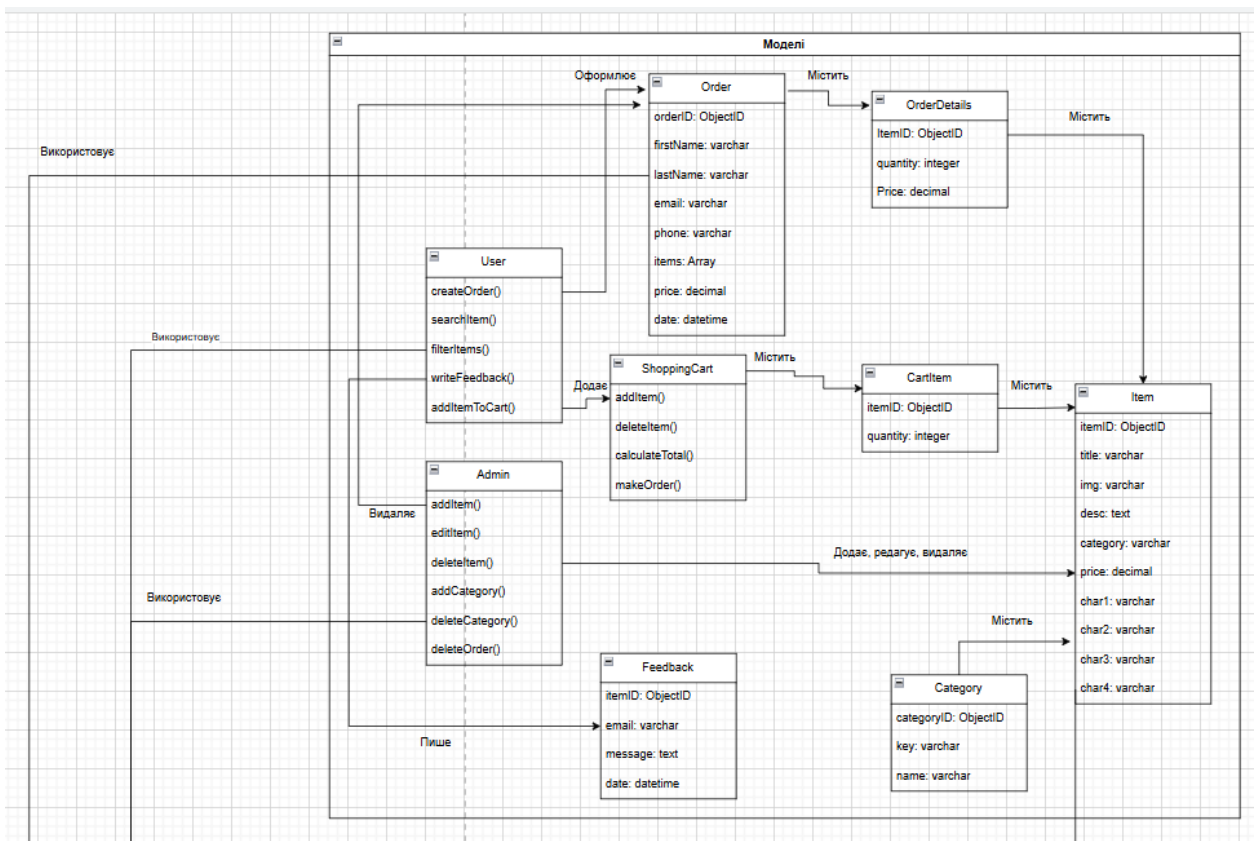


Рис 19. Діаграма класів (моделі)

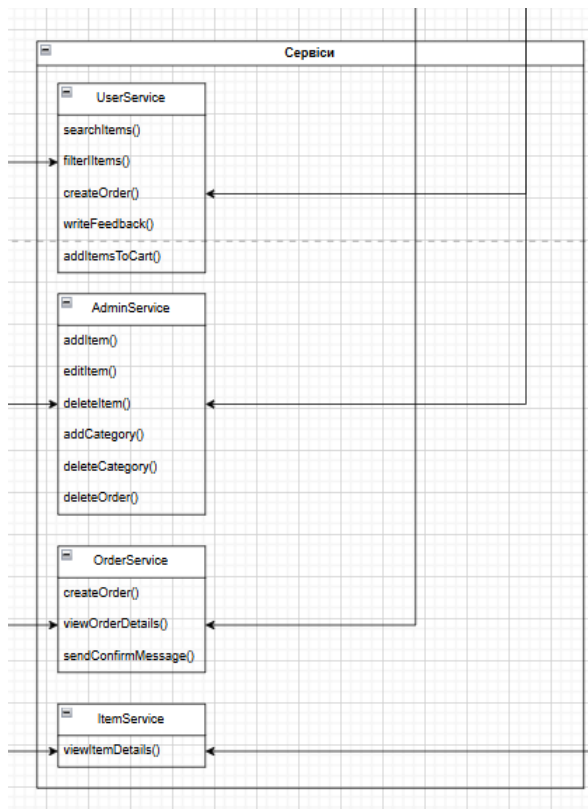


Рис 20. Діаграм класів (сервіси)

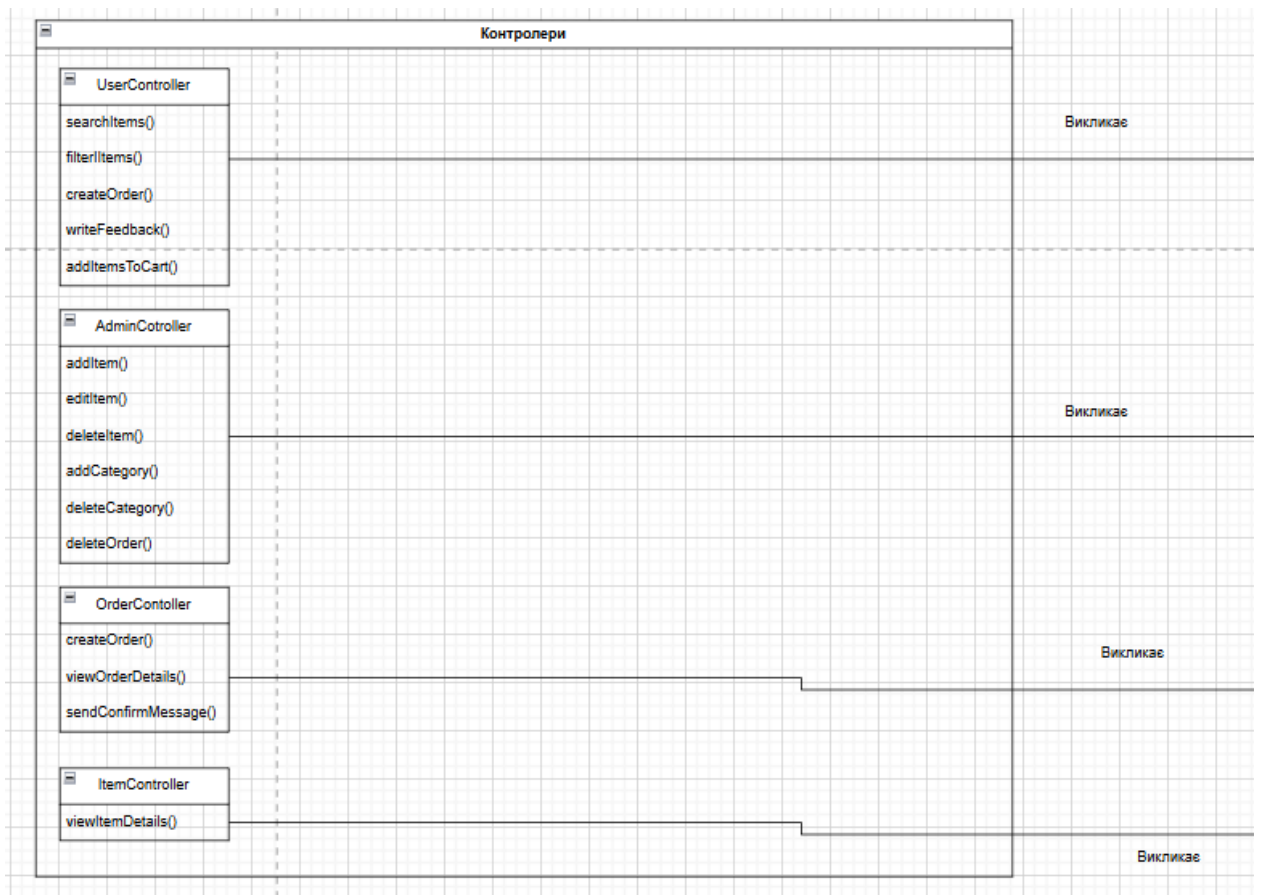


Рис 21. Діаграма класів (контролери)

Висновки

У цій курсовій роботі проведено дослідження та створено веб-орієнтовану систему автоматизації роботи аптеки. Основною метою було вивчення особливостей проєктування та реалізації сучасних веб-систем, а також розробка ефективного рішення, яке відповідатиме потребам як користувачів, так і бізнесу.

Для реалізації серверної і клієнтської частин використано фреймворки React.js та React Router відповідно. Завдяки React.js створено зручний і зрозумілий інтерфейс користувача, а React Router допоміг швидко створити зручну маршрутизацію між усіма сторінками та елементами сайту.

Інтерфейс системи розроблено з урахуванням зручності та простоти використання. Функціонал веб-додатку охоплює можливості перегляду повного каталогу товарів, а також додавання та видалення товарів у кошику, оформлення замовлень, перегляд повної інформації про товар та аптеку на окремих сторінках та форму для зворотнього зв'язку. Увага приділена також функціям фільтрації ліків та їх пошуку за назвою, а також динамічному оновленню даних, що значно покращує користувацький досвід.

Розроблена система може стати корисним інструментом для аптек, які прагнуть автоматизувати процеси, а також для споживачів, яким потрібна зручна платформа замовлення ліків.

Таким чином, у цій роботі було створено та досліджено веб-орієнтовану систему для автоматизації роботи аптеки, яка відповідає сучасним стандартам функціональності, безпеки й масштабованості. Отримані результати можуть бути використані для подальшого удосконалення, розширення функціональних можливостей системи й адаптації її до специфічних запитів бізнесу та кінцевих користувачів.

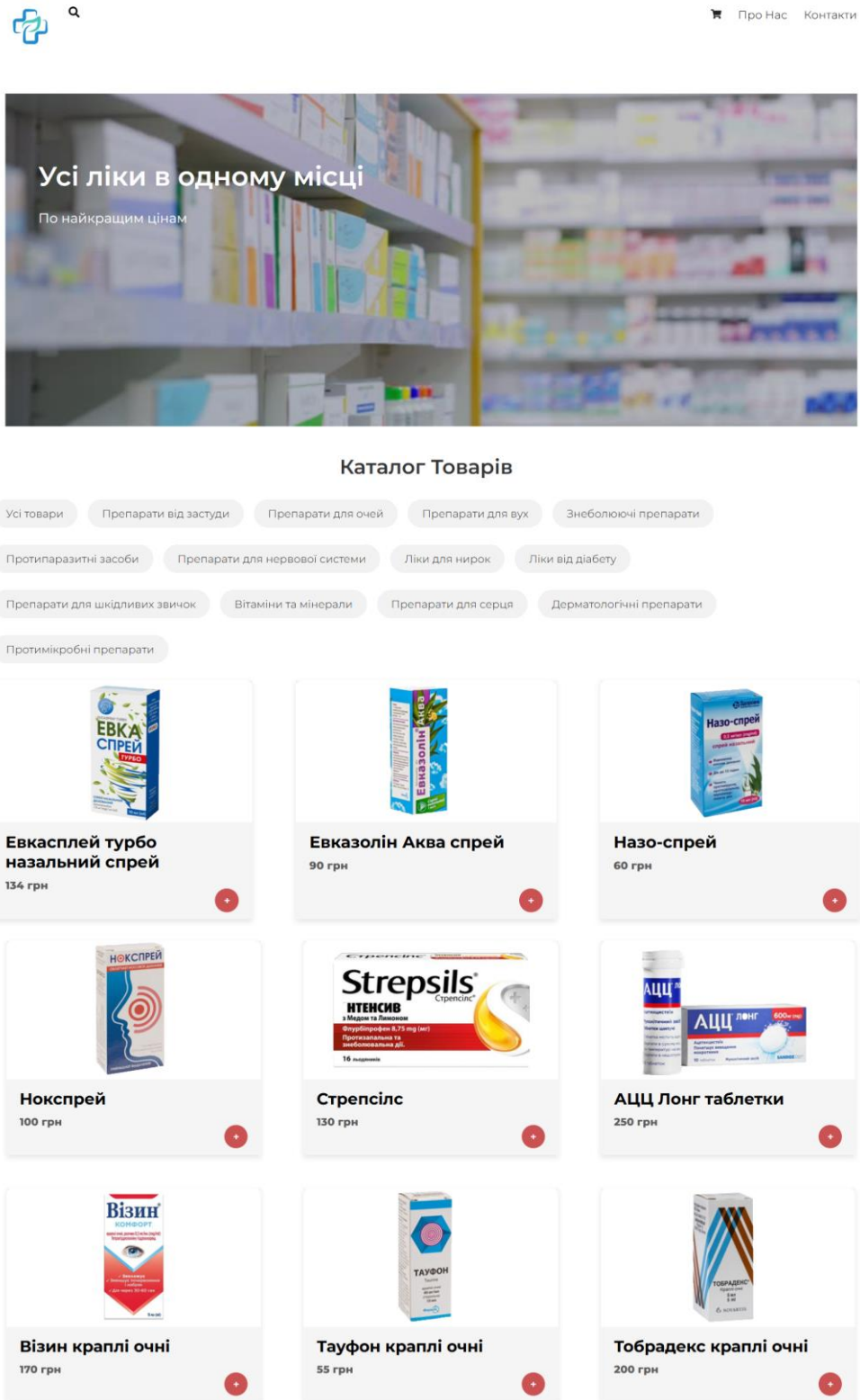
Список використаних джерел



















1. Документація React.js: <https://uk.legacy.reactjs.org/docs/getting-started.html>
2. Документація HTML [<https://developer.mozilla.org/en-US/docs/Web/HTML>]
3. Довідник по HTML і CSS: <https://css.in.ua/>
4. JavaScript документація: <https://devdocs.io/javascript/>
5. Node.js документація: <https://nodejs.org/docs/latest/api/>
6. React Router: <https://reactrouter.com/>
7. React Dom: <https://uk.legacy.reactjs.org/docs/react-dom.html>
8. MongoDB: <https://www.mongodb.com/docs/>
9. E. Brown, Web Development with Node and Express: Leveraging the JavaScript Stack, 2nd ed., O'Reilly Media, 2019, 352 pages.
10. A. Mardan, Pro Express.js: Master Express.js: The Node.js Framework For Your Web Development, Apress, 2014, 308 pages
11. D. Flanagan, JavaScript: The Definitive Guide, 7th ed., O'Reilly Media, 2020, 706 pages
12. "CSS-Tricks: A Complete Guide to Flexbox", Chris Coyier
13. "React Up and Running: Building Web Applications", Stoyan Stefanov
14. "React Design Patterns and Best Practices", Michele Bertoli
15. "CSS Secrets: Better Solutions to Everyday Web Design Problems", Lea Verou
16. Stack Overflow: <https://stackoverflow.com/>

Додатки

Додаток А

Головна сторінка:



 Отипакс краплі вушні 200 грн	 Ототон краплі вушні 170 грн	 Кандибіотик краплі вушні 250 грн
 Лідокаїн 2% розчин для ін'єкцій 45 грн	 Ібупрофен таблетки 60 грн	 Антимігрен-здоров'я таблетки 75 грн
 Плаквеніл таблетки 750 грн	 Бензилбензоату емульсія 30 грн	 Ворміл таблетки 130 грн
 Сонміл таблетки 100 грн	 Карвалол краплі 50 грн	 Флуоксетин таблетки 50 грн
 Лантус Солостар розчин для ін'єкцій 1600 грн	 Табекс Таблетки 450 грн	 Медіхронал-Дарниця гранули 100 грн
 Глутаргін Алкоклін порошок 235 грн	 Кальцію глюконат таблетки 20 грн	 Женьшень настойка 85.50 грн

Ми у соціальних мережах:



Головна Сторінка
Про Нас
Контакти

Quick Link

Contact Us

+380673567890
pharmacylviv@gmail.com
Lviv, Ukraine

Усі права захищені ©

Сторінка товару:

[Про Нас](#)[Контакти](#)

Лідокаїн 2% розчин для ін'єкцій

45 грн

Лідокаїн 2% розчин для ін'єкцій 20 мг/мл ампула 2 мл

[Додати до кошика](#)[Відкрити характеристики](#)

Ми у соціальних мережах:



Quick Link

[Головна Сторінка](#)
[Про Нас](#)
[Контакти](#)

Contact Us

+380673567890
 pharmacylviv@gmail.com
 Lviv, Ukraine

Усі права захищені ©

Сторінка “Про Нас”

[Про Нас](#)[Контакти](#)

Про Нас

Ми молода аптека у самому центрі Львова.
У нас на сайті Ви зможете знайти усі необхідні для Вас ліки по найкращим цінам.



Графік роботи

Пн-Пт: 08:00 - 23:00
Сб: 09:00 - 23:00
Нд: 09:00 - 21:00



Доставка та самовивіз

Доставка виконується кур'єром лише по Львову
Час доставки: 1-3 години з моменту замовлення

Самовивіз можливий із точки видачі при замовленні онлайн через 30 хв після створення замовлення



Оплата

Оплата можлива наступними варіантами:

- Оплата на сайті
- Оплата при отриманні у аптеці

Ми у соціальних мережах:



Quick Link

Головна Сторінка
Про Нас
Контакти

Contact Us

+380673567890
pharmacylviv@gmail.com
Lviv, Ukraine

Усі права захищені ©

Сторінка “Контакти”



🛒 Про Нас Контакти

Зворотній зв'язок

Надіслати

Ми у соціальних мережах:



Quick Link


Головна Сторінка
Про Нас
Контакти

Contact Us

+380673567890
pharmacylviv@gmail.com
Lviv, Ukraine

Результати пошуку:

Результати пошуку за запитом: "Лідо"




Лідокаїн 2% розчин для ін'єкцій


Лідокаїн 2% розчин для ін'єкцій 20 мг/мл ампула 2 мл

45 грн



Кошик


 [Про Нас](#) [Контакти](#)




Евказолін Аква спрей

90 грн

1







Евкасплей турбо назальний спрей

134 грн

2






Тауфон краплі очні

55 грн


1



Загальна вартість: 413.00 грн

Оформити замовлення

Сторінка оформлення замовлення



Оформлення замовлення

Ім'я:


Прізвище:

Email:


Номер телефону:

[Підтвердити замовлення](#)


Ваше замовлення



Евказолін Аква спрей
Кількість: 1
Ціна за шт.: 90 грн



Евкасплей турбо назальний спрей
Кількість: 2
Ціна за шт.: 134 грн



Тауфон краплі очні
Кількість: 1
Ціна за шт.: 55 грн

Загальна вартість: 413.00 грн

Коди програми

Клієнтська частина:

Файл App.js

```
import React, { useState } from "react";
import Header from "../Components/Header";
import Footer from "../Components/Footer";
import Home from "../Pages/Home";
import About from "../Pages/About";
import Contacts from "../Pages/Contacts";
import { BrowserRouter as Router, Route, Routes, Navigate } from "react-router-dom";
import ItemDetails from "../Pages/ItemDetails";
import OrderPage from "../Pages/OrderPage";
import { useNavigate, useLocation } from "react-router-dom";
import SearchResults from "../Pages/SearchResults";
import Toast from "../Components/Toast";
import axios from "axios";
import Items from "../Components/Items";

function App() {
  const location = useLocation();
  const navigate = useNavigate();
  const [orders, setOrders] = React.useState([]);
  const [currentItems, setCurrentItems] = React.useState([]);
  const [items, setItems] = React.useState([]);
  const [cartItems, setCartItems] = useState([]);
  const [toastMessage, setToastMessage] = useState('');

  // Функція для отримання даних з API
  React.useEffect(() => {
    axios.get('http://localhost:5000/api/items')
      .then(res => {
        setItems(res.data); // Оновлюємо стан з отриманими айтемами
      })
      .catch(err => {
        console.error('Помилка при завантаженні айтемів:', err);
      });
  }, []);

  React.useEffect(() => {
    setCurrentItems(items);
  }, [items]);

  const addToOrder = (item) => {
    setCartItems([...cartItems, item]);
    setToastMessage(`${item.title} додано у кошик!`);
  };
}
```

```

const updatedOrders = [...orders];
const existingItem = updatedOrders.find((el) => el._id === item._id);
if (existingItem) {
  existingItem.quantity += 1;
} else {
  updatedOrders.push({ ...item, quantity: 1 });
}
setOrders(updatedOrders);
};

const deleteOrder = (id) => {
  setOrders(orders.filter((el) => el._id !== id));
};

const chooseCategory = (category) => {
  if (category === "all") {
    setCurrentItems(items);
  } else {
    setCurrentItems(items.filter((el) => el.category === category));
  }
};

const clearCart = () => {
  setOrders([]);
}

return (
  <div className="wrapper">
    <Header cartItems={cartItems} orders={orders} onDelete={deleteOrder}
pathname={location.pathname} />
    {toastMessage && (<Toast message={toastMessage} onClose={() =>
setToastMessage('')} />)}
    <Routes>
      <Route path="/" element={<Items items={items} onAdd={addToOrder} />} />
      <Route path="/" element={<Navigate to="/home" />} />
      <Route
        path="/home"
        element={<Home items={currentItems} addToOrder={addToOrder}
chooseCategory={chooseCategory} orders={orders} hideDesc={true} />}
      />
      <Route path="/items/:id" element={<ItemDetails items={items}
addToOrder={addToOrder} />} />
      <Route path="/about" element={<About />} />
      <Route path="/contacts" element={<Contacts />} />
      <Route path="/checkout" element={<OrderPage orders={orders}
clearCart={clearCart} setToastMessage={setToastMessage} navigate={navigate} />}
    />
      <Route path="/search" element={<SearchResults onAddToCart={addToOrder}
/>} />
    </Routes>
  </div>
)

```

```

        {location.pathname !== "/checkout" && <Footer />}
      </div>
    );
  }

export default App;

```

Файл Categories.js:

```

import React, { Component } from 'react';
import axios from 'axios';

export class Categories extends Component {
  constructor(props) {
    super(props);
    this.state = {
      categories: []
    };
  }

  componentDidMount() {
    axios.get('http://localhost:5000/api/categories')
      .then(res => {
        const allCategory = { key: 'all', name: 'Усі товари' };
        this.setState({ categories: [allCategory, ...res.data] });
      })
      .catch(err => {
        console.error('Помилка при завантаженні категорій:', err);
      });
  }

  render() {
    return (
      <div className='categories'>
        {this.state.categories.map(el => (
          <div key={el.key} onClick={() => this.props.chooseCategory(el.key)}>
            {el.name}
          </div>
        ))}
      </div>
    );
  }
}

export default Categories;

```

Файл **index.js**

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { BrowserRouter } from 'react-router-dom';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);
```

Файл **index.css:**

```
@import
url('https://fonts.googleapis.com/css2?family=Montserrat:ital,wght@0,100..900;1,100..900&display=swap');

*{
  margin: 0;
  padding: 0;
}

body{
  background-color: #fff;
  color: #222;
  font-family: 'Montserrat', sans-serif;
  font-weight: 300;
}

.wrapper{
  width: 1280px;
  margin: 50px auto;
}

header{
  position: relative;
}

header .logo_img{
  height: 75px;
  width: 75px;
  margin-right: 20px;
}
```



```

header .shop-cart-button{
  float: right;
  cursor: pointer;
  transition: color, transform 500ms ease;
  margin-top: 10px;
}

header .shop-cart-button:hover,
header .shop-cart-button.active{
  color: #dc3d3d;
  transform: scale(1.5);
}

header .shop-cart{
  position: absolute;
  top: 30px;
  right: 0;
  width: 450px;
  background-color: #fafafa;
  -webkit-box-shadow: 4px 5px 15px -7px #606060;
  box-shadow: 4px 5px 15px -7px #606060;
  z-index: 1000;
  padding: 10px;
  padding-bottom: 0;
}

header .shop-cart .empty h2{
  font-size: 20px;
  margin-bottom: 20px;
}

header .shop-cart .item{
  width: 100%;
  margin-bottom: 10px;
  float: left;
}

header .shop-cart .item img{
  width: 70px;
  float: left;
  margin-right: 20px;
  background-color: #e7e7e7;
}

header .shop-cart .item h2{
  font-size: 20px;
}

header .shop-cart .item b{
  color: #656565;
}

```

```

    font-weight: 600;
}

header .shop-cart .item .quantity-circle{
    background-color: #000000;
    color: #fff;
    font-weight: 600;
    width: 25px;
    height: 25px;
    display: flex;
    align-items: center;
    justify-content: center;
    border-radius: 50%;
    margin: 0 10px;
}

header .shop-cart .item .delete-icon{
    color: #ca5252;
    float: right;
    position: relative;
    bottom: 25px;
    cursor: pointer;
    transition: color, transform 500ms ease;
}

header .shop-cart .item .delete-icon:hover{
    color: #d83030;
    transform: scale(1.5);
}

header .shop-cart .sum{
    float: left;
    width: 100%;
    font-weight: 600;
    font-size: 20px;
    margin-bottom: 10px;
    margin-top: -10px;
}

header .cart-icon-wrapper {
    position: relative;
    display: inline-block;
    float: right;
    margin-top: 2px;
    cursor: pointer;
}

header .cart-count {
    position: absolute;
    top: -5px;

```

```

    right: -10px;
    background-color: #dc3d3d;
    color: white;
    border-radius: 50%;
    padding: 2px 6px;
    font-size: 12px;
    font-weight: bold;
    line-height: 1;
}

header .search-container {
    position: relative;
    display: inline-block;
    transform: translateY(-50px);
    z-index: 1000;
}

header .search-toggle {
    cursor: pointer;
    background: none;
    border: none;
    font-size: 17px;
    margin-bottom: 10px;
    transition: transform 500ms ease;
}

header .search-toggle:hover,
header .search-toggle.active{
    color: #dc3d3d;
    transform: scale(1.25);
}

header .search-dropdown ul {
    position: absolute;
    left: 100%;
    top: 50px;
    right: 0;
    background: white;
    border: 1px solid #ccc;
    padding: 10px;
    width: 250px;
    z-index: 1000;
    box-shadow: 0 4px 10px rgba(0,0,0,0.1);
}

header .search-input {
    top: 20px;
    left: 100%;
    width: 100%;
    padding: 6px;

```

```

margin-bottom: 5px;
border: 1px solid #ccc;
border-radius: 4px;
position: absolute;
right: 0;
width: 250px;
background-color: #fafafa;
-webkit-box-shadow: 4px 5px 15px -7px #606060;
box-shadow: 4px 5px 15px -7px #606060;
z-index: 1000;
padding: 10px;
padding-bottom: 0;
}

header .search-result {
  list-style: none;
  padding: 0;
  margin: 0;
}

header .search-result-item {
  padding: 5px;
  cursor: pointer;
  list-style: none;
}

header .search-result-item:hover,
.search-view-all:hover {
  background-color: #f0f0f0;
}

header .search-result-item-content {
  display: flex;
  align-items: center;
}

header .search-result-item-img {
  width: 50px;
  height: 50px;
  object-fit: cover;
  margin-right: 10px;
}

header .search-result-item h4 {
  font-size: 16px;
  margin: 0;
}

header .search-result-item p {
  margin: 0;
}

```

```

    color: gray;
    font-size: 14px;
}

header .search-view-all {
    text-align: center;
    padding: 10px;
    background-color: #000;
    cursor: pointer;
    list-style: none;
    color: white;
    border-radius: 10px;
    border: none;
    font-size: 16px;
    margin-top: 20px;
    transition: color 500ms ease;
}

header .search-view-all:hover {
    background-color: #d83030;
}

header .complete-order-btn-centralization{
    text-align: center;
}

header .complete-order-btn{
    padding: 10px 20px;
    background-color: #000;
    color: white;
    border-radius: 10px;
    border: none;
    cursor: pointer;
    font-size: 16px;
    margin-top: 20px;
    margin-bottom: 20px;
    transition: color, transform 500ms ease;
}

header .complete-order-btn:hover{
    transform: scale(1.1);
    background-color: #d83030;
}

header .presentation{
    margin: 50px 0;
    background: url('./img/bg.jpg') no-repeat;
    width: 100%;
    height: 500px;
    background-size: cover;
}

```

```

background-position: center center;
background-blend-mode: multiply;
background-color: #bcbcbc;
position: relative;
}

header .nav{
  display: flex;
  list-style: none;
  margin-top: 10px;
}

header .nav.open{
  display: flex;
  flex-direction: column;
  position: absolute;
  top: 60px;
  right: 20px;
  background-color: #fff;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  padding: 10px;
  border-radius: 8px;
  z-index: 10;
}

header ul.nav{
  float: right;
  list-style: none;
}

header ul.nav .nav-li{
  display: inline-block;
  color: #000;
  text-decoration: none;
  margin-left: 25px;
  cursor: pointer;
  transition: opacity 500ms ease;
}

header ul.nav .nav-li:hover{
  opacity: 0.5;
}

header .presentation::after{
  content: 'Усі ліки в одному місці';
  position: absolute;
  top: 100px;
  left: 50px;
  width: 530px;
  font-size: 40px;
}

```

```

    font-weight: 600;
    color: #fff;
}

header .presentation::before{
    content: 'По найкращим цінам';
    position: absolute;
    top: 175px;
    left: 50px;
    width: 530px;
    font-size: 20px;
    font-weight: 400;
    color: #fff;
}

header .catalog-title{
    text-align: center;
    margin-top: 20px;
    margin-bottom: 30px;
    font-size: 30px;
    font-weight: 600;
    color: #333;
}

.categories div{
    display: inline-block;
    background: #f2f2f2;
    border-radius: 50px;
    padding: 10px 20px;
    margin-bottom: 25px;
    margin-right: 15px;
    cursor: pointer;
    border: 1px solid transparent;
    transition: all 500ms ease;
}

.categories div:hover{
    border-color: silver;
    transform: scale(1.1);
}

main{
    display: flex;
    width: 100%;
    flex-wrap: wrap;
    justify-content: space-between;
}

main .item{
    width: 30%;

```

```

margin-bottom: 50px;
background: #f5f5f5;
box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
overflow: hidden;
position: relative;
padding-bottom: 40px;
}

main .item h2 a{
  color: #000;
  font-weight: bold;
  text-decoration: none;
  transition: color 0.3s ease;
}

main .item h2 a:hover {
  color: #d83030;
}

main .item img{
  width: 100%;
  border-radius: 10px 10px 0 0;
  transition: transform 500ms ease;
  background-color: #ffffff;
}

main .item img:hover{
  transform: scale(1.05);
}

main h2,
main p,
main b{
  margin: 10px 20px;
  color: #333;
}

main b{
  color: #595959;
  margin-top: 20px;
  font-weight: 800;
}

main .add-to-cart {
  position: absolute;
  right: 20px;
  bottom: 10px;
  background: #ca2525;
  width: 35px;
  height: 35px;
}

```



```

    text-align: center;
    line-height: 35px;
    color: #fff;
    border-radius: 50%;
    cursor: pointer;
    font-weight: 600;
    transition: transform 500ms ease;
}

main .add-to-cart:hover{
    transform: scale(1.5) translateY(-2.5px);
}

.checkout-btn-container{
    position: fixed;
    bottom: 20px;
    left: 50%;
    transform: translateX(-50%);
}

.checkout-btn{
    background-color: #000;
    color: white;
    padding: 10px 20px;
    border: none;
    font-size: 16px;
    cursor: pointer;
    border-radius: 5px;
}

.checkout-btn:hover{
    background-color: #d83030;
}

footer{
    text-align: center;
    margin-top: 100px;
    background-color: #f1f1f1;
    padding-bottom: 20px;
}

footer p{
    margin-top: 20px;
    font-size: 14px;
    color: #666;
}

footer > div{
    display: flex;
    justify-content: space-between;

```

```

    align-items: flex-start;
    padding: 20px;
    background-color: #f1f1f1;
}

footer .social_media,
footer .navigation,
footer .contact_us{
    flex: 1;
    margin: 0 10px;
}

footer .contact_us{
    display: flex;
    flex-direction: column;
    align-items: flex-start;
    gap: 10px;
}

footer .contact_us div{
    display: flex;
    align-items: center;
    gap: 10px;
}

footer .contact_us .logo_img{
    height: 20px;
    width: 20px;
}

footer h6{
    font-size: 14px;
    margin-bottom: 10px;
    font-weight: 700;
    margin-top: 20px;
}

footer .logo_img{
    height: 30px;
    width: 30px;
    margin-right: 10px;
    transition: transform 0.3s ease;
}

footer .logo_img:hover{
    transform: scale(1.2);
}

footer .nav{
    list-style: none;

```

```

padding: 0;
}

footer .nav li{
margin-bottom: 5px;
}

footer .nav-li{
text-decoration: none;
color: #000;
transition: color 0.3s ease;
font-size: 14px;
}

footer .nav-li:hover{
color: #ca5252;
}

footer span{
display: block;
margin-top: 5px;
font-size: 14px;
}

.page{
padding: 20px;
text-align: center;
}

.page h1{
font-size: 36px;
margin-bottom: 20px;
}

.page p{
font-size: 18px;
color: #000000;
}

.about_us{
padding: 20px;
text-align: center;
}

.about_us h2{
margin-bottom: 20px;
font-size: 30px;
}

.about_us p, li{

```

```

    text-align: left;
    font-size: 17px;
}

.about_us ul{
    margin-top: 10px;
}

.about-us-page-img-1{
    width: 300px;
    height: 300px;
    object-fit: cover;
    display: inline-block;
    margin-top: 40px;
}

.about-us-page-img{
    width: 200px;
    height: 200px;
    object-fit: cover;
    display: inline-block;
    margin-top: 40px;
}

.contacts{
    text-align: center;
    margin-top: 50px;
}

.contacts h1{
    margin-bottom: 20px;
}

.contacts .form-control{
    width: 100%;
    padding: 12px;
    margin-bottom: 16px;
}

.contacts .form-control::placeholder{
    font-size: 16px;
    color: #888;
}

.contacts .btn-submit{
    font-size: 20px;
    padding: 12px 24px;
    background-color: #000;
    border: none;
    color: #fff;
}

```

```

    font-weight: bold;
    border-radius: 5px;
    transition: all 0.5s ease;
}

.contacts .btn-submit:hover{
    transform: scale(1.1);
    background-color: #d83030;
}

.contact-status {
    margin-top: 15px;
    font-weight: bold;
}

.item-details-container{
    width: 100%;
    height: 100%;
    margin: 50px auto;
    padding: 20px;
    border-radius: 8px;
    display: flex;
    justify-content: center;
}

.item-details{
    display: flex;
    gap: 20px;
}

.item-image img{
    width: 500px;
    height: 450px;
    border-radius: 8px;
    box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
    background-color: #e7e7e7;
    margin-right: 100px;
}

.item-info{
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    gap: 20px;
    margin-right: 150px;
}

.item-info h2{
    font-size: 34px;
    font-weight: 600;
}

```

```

    margin-top: 20px;
}

.item-price{
    font-size: 20px;
    font-weight: 700;
    color: #000;
}

.item-desc{
    font-size: 16px;
    color: #383838;
    line-height: 1.6;
}

.add-to-cart-btn{
    background-color: #000;
    color: #fff;
    font-size: 16px;
    font-weight: 600;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease, transform 0.3s ease;
    width: 300px;
    margin-top: 20px;
    margin-bottom: 20px;
}

.add-to-cart-btn:hover{
    background-color: #d83030;
    transform: scale(1.05);
}

.char-toogle-container{
    position: relative;
}

.item-char-container{
    max-height: 0;
    overflow: hidden;
    transform: translateY(-200px);
    opacity: 0;
    transition: transform 0.5s ease, opacity 0.5s ease, max-height 0.5s ease;
}

.item-char-container.open{
    max-height: 200px;
    transform: translateY(0);
}

```

```

    opacity: 1;
}

.toogle-char-btn{
    color: blue;
    cursor: pointer;
    margin-bottom: 15px;
    font-size: 16px;
    transition: color 0.3s ease;
}

.toogle-char-btn:hover{
    color: rgb(134, 134, 255);
}

.item-char{
    opacity: 0;
    transition: opacity 0.3s ease;
}

.item-char.open{
    opacity: 1;
}

.item-char .char-list{
    list-style-type: none;
    padding: 0;
    margin: 0;
}

.item-char .char-list-item{
    font-size: 16px;
    line-height: 1.5;
    border-bottom: 1px solid #ccc;
    padding: 5px 0;
}

.item-char .char-list-item:last-child{
    border-bottom: none;
}

.checkout-page{
    display: flex;
    justify-content: space-between;
    padding: 20px;
}

.checkout-form{
    flex: 1;
    margin-right: 20px;
}

```

```

border: 1px solid #ddd;
padding: 20px;
border-radius: 5px;
background: #f9f9f9;
}

.checkout-form form{
  margin-top: 10px;
}

.checkout-form form label{
  display: block;
  margin-bottom: 10px;
}

.checkout-form form input{
  width: 100%;
  padding: 8px;
  margin-top: 5px;
  margin-bottom: 15px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.checkout-cart{
  flex: 1;
  border: 1px solid #ddd;
  padding: 20px;
  border-radius: 5px;
  background: #f9f9f9;
}

.checkout-cart img{
  width: 100px;
  height: 50px;
  margin-right: 10px;
  vertical-align: middle;
}

.order-summary-item{
  display: flex;
  align-items: center;
  margin-bottom: 10px;
  margin-top: 10px;
}

.checkout-total{
  font-weight: bold;
  margin-top: 20px;
}

```



```

.search-result-page {
  padding: 20px;
}

.search-result-page h2 {
  margin-bottom: 20px;
}

.toast {
  position: fixed;
  top: -50px;
  right: 20px;
  background-color: #333;
  color: #fff;
  padding: 10px 20px;
  border-radius: 10px;
  z-index: 1000;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
  animation: fadeIn 0.3s ease-in-out forwards;
  overflow: hidden;
}

@keyframes fadeIn {
  0% {
    top: -50px;
    opacity: 0;
  }
  100% {
    top: 20px;
    opacity: 1;
  }
}

.toast.fade-out {
  animation: fadeOut 0.15s ease-in-out forwards;
}

@keyframes fadeOut {
  0% {
    opacity: 1;
    top: 20px;
  }
  100% {
    opacity: 0;
    top: -50px;
  }
}

.toast .progress-bar {

```

```

position: absolute;
bottom: 0;
left: 0;
height: 4px;
background-color: #4caf50;
width: 100%;
animation: shrink 2s linear forwards;
border-bottom-left-radius: 10px;
border-bottom-right-radius: 10px;
}

@keyframes shrink {
  from {
    width: 100%;
  }
  to {
    width: 0%;
  }
}

```

Папка Components

Файл Footer.js

```

import React from 'react'
import {Link} from 'react-router-dom';
import insta_logo from '../img/instagram.png'
import phone_logo from '../img/phone.png'
import mail_logo from '../img/mail.png'
import location_logo from '../img/location.png'

export default function Footer() {
  return (
    <footer>
      <div>
        <div className='social_media'>
          <h6>Ми у соціальних мережах:</h6>
          <a href='https://www.instagram.com' target='_blank' rel='noopener
noreferrer'>
            <img className="logo_img" src={insta_logo} alt="Логотип" />
          </a>
        </div>
        <div className='navigation'>
          <h6>Quick Link</h6>
          <ul className="nav">
            <li><Link className='nav-li' to="/home">Головна Сторінка</Link></li>
            <li><Link className="nav-li" to="/about">Про Нас</Link></li>
            <li><Link className="nav-li" to="/contacts">Контакти</Link></li>
          </ul>
        </div>
      </div>
    </footer>
  )
}

```

```

    <div className='contact_us'>
      <h6>Contact Us</h6>
      <div>
        <img className="logo_img" src={phone_logo} alt="Логотип" />
        <span>+380673567890</span>
      </div>
      <div>
        <img className="logo_img" src={mail_logo} alt="Логотип" />
        <span>pharmacylviv@gmail.com</span>
      </div>
      <div>
        <img className="logo_img" src={location_logo} alt="Логотип" />
        <span>Lviv, Ukraine</span>
      </div>
    </div>
  </div>
  <p>Усі права захищені &copy;</p>
</footer>
)
}

```

Файл Header.js

```

import React, { useState, useEffect, useRef } from 'react';
import { Link, useLocation, useNavigate } from 'react-router-dom';
import { FaShoppingCart, FaSearch } from 'react-icons/fa';
import Order from './Order';
import logo from '../img/logo.png';

const showOrder = (props) => {
  let sum = 0;
  props.orders.forEach(el => sum += Number.parseFloat(el.price) * el.quantity);
  return (
    <div>
      {props.orders.map(el => (
        <Order onDelete={props.onDelete} key={el._id} item={el} />
      ))}
      <p className="sum">Загальна вартість: {sum.toFixed(2)} грн</p>
      <div className="complete-order-btn-centralization">
        <Link to="/checkout">
          <button className="complete-order-btn">Оформити замовлення</button>
        </Link>
      </div>
    </div>
  );
};

const showNothing = () => (
  <div className="empty">

```

```

    <h2>Корзина порожня</h2>
  </div>
);

export default function Header(props) {
  const [cartOpen, setCartOpen] = useState(false);
  const [searchOpen, setSearchOpen] = useState(false);
  const [query, setQuery] = useState('');
  const [results, setResults] = useState([]);
  const navigate = useNavigate();
  const location = useLocation();
  const searchRef = useRef(null);
  const cartRef = useRef(null);

  useEffect(() => {
    const fetchSearchResults = async () => {
      if (query.trim() === '') {
        setResults([]);
        return;
      }

      try {
        const response = await
fetch(`http://localhost:5000/api/items/search?q=${query}`);
        const data = await response.json();
        setResults(data.slice(0, 3));
      } catch (err) {
        console.error("Помилка пошуку: ", err);
      }
    };

    const timeout = setTimeout(fetchSearchResults, 300);
    return () => clearTimeout(timeout);
  }, [query]);

  useEffect(() => {
    if (location.pathname.includes('/product/')) {
      setSearchOpen(false);
      setQuery('');
      setResults([]);
    }
  }, [location.pathname]);

  useEffect(() => {
    const handleClickOutside = (event) => {
      if (searchRef.current && !searchRef.current.contains(event.target)) {
        setSearchOpen(false);
      }
    }

    // Додаємо перевірку для кліку поза корзиною

```

```

    if (cartRef.current && !cartRef.current.contains(event.target)) {
      setCartOpen(false);
    }
  };

  if (searchOpen || cartOpen) {
    document.addEventListener('mousedown', handleClickOutside);
  } else {
    document.removeEventListener('mousedown', handleClickOutside);
  }

  return () => {
    document.removeEventListener('mousedown', handleClickOutside);
  };
}, [searchOpen, cartOpen]);

return (
  <header>
    <div>
      <Link to="/home">
        <img className="logo_img" src={logo} alt="Логотип" />
      </Link>
      {location.pathname !== '/checkout' && (
        <>
          <ul className="nav">
            <li><Link className="nav-li" to="/about">Про Нас</Link></li>
            <li><Link className="nav-li" to="/contacts">Контакти</Link></li>
          </ul>
          <div className="cart-icon-wrapper" onClick={() =>
setCartOpen(!cartOpen)}>
            <FaShoppingCart
              className={`shop-cart-button ${cartOpen && 'active'}`} />
            {props.orders.length > 0 && (
              <span className="cart-count">{props.orders.reduce((acc, item) =>
acc + item.quantity, 0)}</span>
            )}
          </div>
          <div className="search-container" ref={searchRef}>
            <button className="search-toggle" onClick={() =>
setSearchOpen(!searchOpen)}>
              <FaSearch />
            </button>
            {searchOpen && (
              <div className="search-dropdown">
                <input
                  type="text"
                  placeholder="Пошук товарів..."
                  value={query}
                  onChange={(e) => setQuery(e.target.value)}

```

```

        className="search-input"
      />
    {results.length > 0 && (
      <ul className="search-result">
        {results.map((item) => (
          <li
            key={item._id}
            className="search-result-item"
            onClick={() => {
              setSearchOpen(false);
              setQuery('');
              setResults([]);
              navigate(`/items/${item._id}`);
            }}
          >
            <div className="search-result-item-content">
              <img
                src={item.img}
                alt={item.title}
                className="search-result-item-img"
              />
              <div>
                <h4>{item.title}</h4>
                <p>{item.price} грн</p>
              </div>
            </div>
          </li>
        ))}
        <li
          className="search-view-all"
          onClick={() => {
            const searchQuery = query;
            setSearchOpen(false);
            setQuery('');
            setResults([]);
            navigate(`/search?q=${searchQuery}`);
          }}
        >
          Переглянути всі результати
        </li>
      </ul>
    )}
  </div>
)}
</div>

{cartOpen && (
  <div className="shop-cart" ref={cartRef}>
    {props.orders.length > 0 ? showOrder(props) : showNothing()}
  </div>
)}

```

```

    })
  </>
  })
</div>

{location.pathname === '/home' && (
  <>
    <div className="presentation"></div>
    <div className="catalog-title">Каталог Товарів</div>
  </>
)}
</header>
);
}

```

Файл Item.js

```

import React from 'react'
import { Link } from 'react-router-dom'

export const Item = ({item, onAdd, hideDesc}) => {
  return (
    <div className='item'>
      <Link to={`/${items}/${item._id}`}>
        <img src={item.img} alt={item.title}/>
      </Link>
      <h2>
        <Link to={`/${items}/${item._id}`}>{item.title}</Link>
      </h2>
      {!hideDesc && <p>{item.desc}</p>}
      <b>{item.price} грн</b>
      <div className='add-to-cart' onClick={() => onAdd(item)}></div>
    </div>
  )
}

export default Item

```

Файл Items.js

```

import React, { Component } from 'react';
import Item from './Item';

export class Items extends Component {
  render() {
    return (

```

```

    <main>
      {this.props.items.map(el => (
        <Item key={el.id} item={el} onAdd={this.props.onAdd}
hideDesc={this.props.hideDesc} />
      ))}
    </main>
  );
}
}

export default Items;

```

Файл Order.js

```

import React, { Component } from 'react'
import { FaTrash } from 'react-icons/fa'

export class Order extends Component {
  render() {
    return (
      <div className='item'>
        <img src={this.props.item.img} alt={this.props.item.title}/>
        <h2>{this.props.item.title}</h2>
        <b>{this.props.item.price} грн</b>
        <div className='quantity-circle'>{this.props.item.quantity}</div>
        <FaTrash className='delete-icon' onClick={() =>
this.props.onDelete(this.props.item._id)}>/>
      </div>
    )
  }
}

export default Order

```

Файл Toast.js

```

import React, { useEffect, useState } from "react";

const Toast = ({ message, onClose }) => {
  const [show, setShow] = useState(true);

  useEffect(() => {
    const timer = setTimeout(() => {
      setShow(false);
      setTimeout(onClose, 150);
    }, 2000);
    return () => clearTimeout(timer);
  }, [onClose]);

```



```

    }, [onClose]));

    return (
      <div className={`toast ${!show ? "fade-out" : ""}`}>
        {message}
        <div className="progress-bar"></div>
      </div>
    );
  }
}

export default Toast;

```

Папка Pages

Файл About.js

```

import React, { Component } from 'react';
import About_Us from '../img/about_img/about_us.jpg'
import Job_Time from '../img/about_img/job_time.png'
import Payment from '../img/about_img/payment.png'
import Shipping from '../img/about_img/shipping.png'

export default class About extends Component {
  render() {
    return (
      <div className='about_us'>
        <img src={About_Us} className='about-us-page-img-1' />
        <h2>Про Нас</h2>
        <p>Ми молода аптека у самому центрі Львова.</p>
        <p>У нас на сайті Ви зможете знайти усі необхідні для Вас ліки по
найкращим цінам.</p>
        <img src={Job_Time} className='about-us-page-img' />
        <h2>Графік роботи</h2>
        <p><strong>Пн-Пт:</strong> 08:00 - 23:00</p>
        <p><strong>Сб:</strong> 09:00 - 23:00</p>
        <p><strong>Нд:</strong> 09:00 - 21:00</p>
        <img src={Shipping} className='about-us-page-img' />
        <h2>Доставка та самовивіз</h2>
        <p>Доставка виконується кур'єром лише по Львову</p>
        <p><strong>Час доставки:</strong> 1-3 години з моменту
замовлення</p>
        <br></br>
        <p>Самовивіз можливий із точки видачі при замовленні онлайн
<strong>через 30 хв після створення замовлення</strong></p>
        <img src={Payment} className='about-us-page-img' />
        <h2>Оплата</h2>
        <p><strong>Оплата можлива наступними варіантами:</strong></p>
        <ul>

```

```

        <li>Оплата на сайті</li>
        <li>Оплата при отриманні у аптеці</li>
      </ul>
    </div>
  );
}
}

```

Файл **Contacts.js**

```

import React, { Component } from 'react';
import { Container, Form, Button } from 'react-bootstrap';
import Toast from '../Components/Toast';

export default class Contacts extends Component {
  constructor(props) {
    super(props);
    this.state = {
      email: '',
      message: '',
      showToast: false,
      toastMessage: ''
    };
  }

  handleSubmit = async (e) => {
    e.preventDefault();

    try {
      const response = await fetch('http://localhost:5000/api/feedback', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
        },
        body: JSON.stringify({
          email: this.state.email,
          message: this.state.message
        })
      });
    }

    const data = await response.json();

    if (response.ok) {
      this.setState({
        email: '',
        message: '',
        toastMessage: data.message || 'Повідомлення надіслано!',
        showToast: true
      });
    }
  }
}

```

```

    } else {
      this.setState({
        toastMessage: data.error || 'Сталася помилка',
        showToast: true
      });
    }
  } catch (error) {
    this.setState({
      toastMessage: 'Помилка з'єднання з сервером',
      showToast: true
    });
  }
};

render() {
  return (
    <Container className="contacts">
      <h1>Зворотній зв'язок</h1>
      <Form onSubmit={this.handleSubmit}>
        <Form.Group controlId="formBasicEmail">
          <Form.Control
            className="form-control"
            type="email"
            placeholder="Ваш email"
            value={this.state.email}
            onChange={(e) => this.setState({ email: e.target.value })}
            required
          />
        </Form.Group>
        <Form.Group controlId="formBasicMessage">
          <Form.Control
            className="form-control"
            as="textarea"
            rows="3"
            placeholder="Напишіть ваше питання"
            value={this.state.message}
            onChange={(e) => this.setState({ message: e.target.value })}
            required
          />
        </Form.Group>
        <Button className="btn-submit" variant="primary" type="submit">
          Надіслати
        </Button>
      </Form>

      {this.state.showToast && (
        <Toast
          message={this.state.toastMessage}
          onClose={() => this.setState({ showToast: false })}
        />
      )}
    </Container>
  );
}

```

```

    })
  </Container>
);
}
}

```

Файл Home.js

```

import React from 'react';
import Items from '../Components/Items';
import Categories from '../Categories'

const Home = ({ items, addToOrder, chooseCategory, hideDesc }) => (
  <div>
    <Categories chooseCategory={chooseCategory} />
    <Items items={items} onAdd={addToOrder} hideDesc={hideDesc}/>
  </div>
);

export default Home;

```

Файл ItemDetails.js

```

import React, { useState } from "react";
import { useParams } from "react-router-dom";

const ItemDetails = ({items, addToOrder}) => {
  const {id} = useParams()
  console.log('ID з URL:', id);
  console.log('items:', items);
  const item = items.find(item => item._id === id)
  const [isCharVisible, setIsCharVisible] = useState(false)

  if(!item) {
    return <h2>Item not found</h2>
  }

  const handleAddToCart = () => {
    addToOrder(item);
  };

  const toggleCharVisibility = () => {
    setIsCharVisible(!isCharVisible)
  }

  return (

```

```

    <div className="item-details-container">
      <div className="item-details">
        <div className="item-image">
          <img src={item.img} alt={item.title}></img>
        </div>
        <div className="item-info">
          <h2>{item.title}</h2>
          <b className="item-price">{item.price} грн</b>
          <p className="item-desc">{item.desc}</p>
          <button className="add-to-cart-btn"
onClick={handleAddToCart}>Додати до кошика</button>
          <p className="toggle-char-btn"
onClick={toggleCharVisibility}>{isCharVisible ? "Закрити Характеристики" :
"Відкрити характеристики"}</p>
          <div className={`item-char ${isCharVisible ? "open" : ""}`}>
            <ul className="char-list">
              <li className="char-list-item">{item.char1}</li>
              <li className="char-list-item">{item.char2}</li>
              <li className="char-list-item">{item.char3}</li>
              <li className="char-list-item">{item.char4}</li>
            </ul>
          </div>
        </div>
      </div>
    </div>
  )
}

export default ItemDetails;

```

Файл OrderPage.js

```

import React, { useState } from 'react';

export default function OrderPage({ orders, clearCart, setToastMessage, navigate
}) {
  const [formData, setFormData] = useState({
    firstName: '',
    lastName: '',
    email: '',
    phone: '',
  });

  let sum = 0;
  orders.forEach(el => (sum += Number.parseFloat(el.price) * el.quantity));

  const handleInputChange = (e) => {
    setFormData({
      ...formData,

```

```

        [e.target.name]: e.target.value
    });
};

const handleSubmit = async (e) => {
    e.preventDefault();

    const formData = new FormData(e.target);
    const orderData = {
        firstName: formData.get('firstName'),
        lastName: formData.get('lastName'),
        email: formData.get('email'),
        phone: formData.get('phone'),
        items: orders,
        total: sum.toFixed(2)
    };

    try {
        const response = await fetch('http://localhost:5000/api/orders', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(orderData)
        });

        if (!response.ok) {
            const errorData = await response.json();
            throw new Error(errorData.message || 'Помилка при створенні замовлення!');
        }

        setToastMessage('Замовлення успішно оформлено!');
        clearCart();
        navigate('/home');
    } catch (error) {
        console.error(error.message);
        setToastMessage('Не вдалося створити замовлення');
    }
};

return (
    <div className='checkout-page'>
        <div className='checkout-form'>
            <h2>Оформлення замовлення</h2>
            <form onSubmit={handleSubmit}>
                <label>
                    Ім'я:
                    <input type='text' name='firstName'
value={formData.firstName} onChange={handleInputChange} required />
                </label>
                <label>

```

```

        Прізвище:
        <input type='text' name='lastName' value={formData.lastName}
onChange={handleInputChange} required />
      </label>
      <label>
        Email:
        <input type='email' name='email' value={formData.email}
onChange={handleInputChange} required />
      </label>
      <label>
        Номер телефону:
        <input type='tel' name='phone' value={formData.phone}
onChange={handleInputChange} required />
      </label>
      <button type='submit'>Підтвердити замовлення</button>
    </form>
  </div>
  <div className='checkout-cart'>
    <h2>Ваше замовлення</h2>
    {orders.map(el => (
      <div key={el.id} className='order-summary-item'>
        <img src={el.img} alt={el.title} />
        <div>
          <h3>{el.title}</h3>
          <p>Кількість: {el.quantity}</p>
          <p>Ціна за шт.: {el.price} грн</p>
        </div>
      </div>
    ))}
    <p className='checkout-total'>Загальна вартість: {sum.toFixed(2)}
    грн</p>
  </div>
</div>
);
}

```

Файл Categories.js

```

// SearchResults.js
import React, { useEffect, useState } from 'react';
import { useLocation } from 'react-router-dom';
import Items from '../Components/Items';

function useQuery() {
  return new URLSearchParams(useLocation().search);
}

export default function SearchResults({ onAddToCart }) {
  const query = useQuery().get('q') || '';

```

```

const [results, setResults] = useState([]);

useEffect(() => {
  const fetchResults = async () => {
    if (query.trim() === '') {
      setResults([]);
      return;
    }

    try {
      const res = await
fetch(`http://localhost:5000/api/items/search?q=${query}`);
      const data = await res.json();
      setResults(data);
    } catch (err) {
      console.error('Помилка при завантаженні результатів пошуку:', err);
    }
  };

  fetchResults();
}, [query]);

return (
  <div className="search-result-page">
    <h2>
      Результати пошуку за запитом: "{query}"
    </h2>
    <Items items={results} onAdd={onAddToCart} hideDesc={false} />
  </div>
);
}

```

Серверна частина:

Файл **server.js**

```

const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const dotenv = require('dotenv');

dotenv.config();
const app = express();

app.use(cors());
app.use(express.json({limit: '2mb'}));

```



```

const itemRoutes = require('./routes/items');
const categoryRoutes = require('./routes/categories');
const feedbackRoutes = require('./routes/feedback');
const orderRoutes = require('./routes/orders');

app.use('/api/items', itemRoutes);
app.use('/api/categories', categoryRoutes);
app.use('/api/feedback', feedbackRoutes);
app.use('/api/orders', orderRoutes);

mongoose.connect(process.env.MONGO_URI, { useNewUrlParser: true,
useUnifiedTopology: true })
  .then(() => {
    console.log('MongoDB connected');
    app.listen(process.env.PORT, () => console.log(`Server started on port
${process.env.PORT}`));
  })
  .catch(err => console.error(err));

```

Папка **models**

Файл **Category.js**

```

const mongoose = require('mongoose');

const categorySchema = new mongoose.Schema({
  key: String,
  name: String
});

module.exports = mongoose.model('Category', categorySchema);

```

Файл **Feedback.js**

```

const mongoose = require('mongoose');

const FeedbackSchema = new mongoose.Schema({
  email: {type: String, required: true},
  message: {type: String, required: true},
  date: {type: Date, default: Date.now}
});

module.exports = mongoose.model('Feedback', FeedbackSchema);

```

Файл **Item.js**

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

const itemSchema = new mongoose.Schema({
  title: String,
  img: String,
  desc: String,
  category: String,
  price: String,
  char1: String,
  char2: String,
  char3: String,
  char4: String
});

module.exports = mongoose.models.Item || mongoose.model('Item', itemSchema);
```

Файл **Order.js**

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

const orderSchema = new mongoose.Schema({
  firstName: {type: String, required: true},
  lastName: {type: String, required: true},
  email: {type: String, required: true},
  phone: {type: String, required: true},
  items: [
    {
      title: String,
      quantity: Number,
      price: Number,
      img: String
    }
  ],
  total: {type: Number, required: true},
  date: {type: Date, default: Date.now}
});

module.exports = mongoose.model('Order', orderSchema);
```

Папка **routes**

Файл **categories.js**

```
const express = require('express');
const router = express.Router();
const Category = require('../models/Category');

// Отримати всі категорії
router.get('/', async (req, res) => {
  try {
    const categories = await Category.find();
    res.json(categories);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

// Додати нову категорію
router.post('/', async (req, res) => {
  const category = new Category(req.body);
  try {
    const newCategory = await category.save();
    res.status(201).json(newCategory);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
});

module.exports = router;
```

Файл **feedback.js**

```
const express = require('express');
const router = express.Router();
const Feedback = require('../models/Feedback');

router.post('/', async (req, res) => {
  try{
    const {email, message} = req.body;
    const feedback = new Feedback({ email, message });
    await feedback.save();
    res.status(201).json({ message: 'Повідомлення надіслано!'});
  } catch (err) {
    res.status(500).json({ error: 'Помилка збереження повідомлення'});
  }
});

module.exports = router;
```

Файл items.js

```
const express = require('express');
const router = express.Router();
const Item = require('../models/Item');

// Отримати всі товари
router.get('/', async (req, res) => {
  try {
    const items = await Item.find();
    res.json(items);
  } catch (err) {
    res.status(500).json({ message: err.message });
  }
});

// Додати новий товар
router.post('/', async (req, res) => {
  const item = new Item(req.body);
  try {
    const newItem = await item.save();
    res.status(201).json(newItem);
  } catch (err) {
    res.status(400).json({ message: err.message });
  }
});

// Пошук товарів за запитом
router.get('/search', async (req, res) => {
  const query = req.query.q;
  if (!query) {
    return res.status(400).json({ message: 'Пошуковий запит не може бути порожнім' });
  }

  try {
    // Використовуємо регулярний вираз для пошуку по полю title
    const items = await Item.find({
      title: { $regex: query, $options: 'i' }, // Пошук без врахування регістру
    }).limit(10); // Лімітуємо результати до 10 товарів

    res.json(items);
  } catch (err) {
    console.error("Помилка пошуку: ", err);
    res.status(500).json({ message: 'Помилка сервера' });
  }
});

module.exports = router;
```

Файл orders.js

```
const express = require('express');
const router = express.Router();
const Order = require('../models/Order');
const transporter = require('../utils/mail');

router.post('/', async (req, res) => {
  const {firstName, lastName, email, phone, items, total} = req.body;

  const newOrder = new Order ({
    firstName,
    lastName,
    email,
    phone,
    items,
    total
  });

  try {
    await newOrder.save();

    const itemList = items.map(item => `<li>${item.title} (x${item.quantity})
- ${item.price} грн </li>`).join('');

    const mailOptions = {
      from: process.env.EMAIL_USER,
      to: email,
      subject: 'Підтвердження замовлення',
      html: `
        <h2>Дякуємо за ваше замовлення, ${firstName}! </h2>
        <p>Ми отримали ваше замовлення та вже почали формувати його. </p>
        <h3>Деталі замовлення:</h3>
        <ul>${itemList}</ul>
        <p><strong>Загальна сума: </strong>${total} грн</p>
        <p>З повагою, Аптечна лавка</p>
      `
    };

    await transporter.sendMail(mailOptions);

    res.status(201).json({message: 'Замовлення успішно створено'});
  } catch (error) {
    res.status(500).json({ message: 'Сталася помилка при створенні
замовлення'});
  }
});

module.exports = router;
```

Папка **utils**

Файл **mail.js**

```
// utils/mail.js
const nodemailer = require('nodemailer');

const transporter = nodemailer.createTransport({
  service: 'gmail',
  auth: {
    user: process.env.EMAIL_USER,
    pass: process.env.EMAIL_PASS,
  },
});

module.exports = transporter;
```