

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій
(повне найменування факультету)

Комп'ютерні системи та мережі
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалаврський
(ступінь вищої освіти)

на тему: РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ІДЕНТИФІКАЦІЇ ТА
ВІДОБРАЖЕННЯ СКЛАДНИКІВ ТОВАРІВ

Виконав(ла): студент(ка) 4 курсу,
групи КНТ-513сп

Спеціальності
123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЛАЗЬКО О.В.
(ПРИЗВИЩЕ та ініціали)

Керівник ТЯГУНОВА М.Ю.
(ПРИЗВИЩЕ та ініціали)

Рецензент РОМАНОВСЬКИЙ О.В.
(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет Комп'ютерних наук і технологій
Кафедра комп'ютерних систем та мереж
Ступінь вищої освіти бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри КУДЕРМЕТОВ Р.К.

«14» квітня 2026 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Лазька Олександра Владиславовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка інтелектуальної системи ідентифікації та відображення складників товарів

керівник проєкту (роботи) к.т.н., доцент, ТЯГУНОВА М.Ю.

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

затверджені наказом закладу вищої освіти від « 08 » квітня 2026 року № 151

2. Строк подання студентом проєкту (роботи) 01.06.2026 р.

3. Вихідні дані до проєкту (роботи) опис предметної області, технології розробки клієнтської та серверної частини

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз предметної області;

2) Проєктування системи;

3) Реалізація системи;

4) Проєктування інтерфейсу користувача;

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

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	ТЯГУНОВА М.Ю.		
Нормоконтроль	ГОЛУБ Т.В.		

7. Дата видачі завдання «14» квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	до 18.04.2026	
2	Аналіз існуючих проблем та подібних рішень	до 20.04.2026	
3	Постановка задачі дипломного проєкту	до 22.04.2026	
4	Проектування загальної структури системи	до 24.04.2026	
5	Визначення функціональних та нефункціональних вимог	до 27.04.2026	
6	Опис функціонування системи та алгоритму її роботи	до 02.05.2026	
7	Проектування архітектури програмного забезпечення	до 07.05.2026	
8	Проектування бази даних та взаємодії компонентів	до 12.05.2026	
9	Реалізація системи та інтеграція сервісу штучного інтелекту	до 18.05.2026	
10	Проектування інтерфейсу користувача та Тестування системи	до 22.06.2026	
11	Оформлення пояснювальної записки та Проходження нормоконтролю	до 01.06.2026	
12	Проходження рецензування та перевірки на академічний плагіат	до 10.06.2026	

Студент(ка)

_____ (підпис)

Олександр ЛАЗЬКО

(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ (підпис)

Марія ТЯГУНОВА

(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалаврського диплому:
73 с., 4 таблиці, 8 рисунків, 2 додатки, 15 джерел.

ІНФОРМАЦІЙНА СИСТЕМА, ШТРИХ-КОД, СКЛАДНИКИ ПРОДУКТУ,
БАЗА ДАНИХ, ШТУЧНИЙ ІНТЕЛЕКТ, ВЕБЗАСТОСУНОК, REACT, NODE.JS,
SQLITE, GEMINI API.

Об'єктом розробки є інтелектуальна система для ідентифікації та відображення складників товарів.

Метою цієї роботи є спрощення процесу визначення складників продукції та ідентифікації товарів за рахунок використання інтелектуальних технологій.

У першому розділі аналізується предметна область, аналогічні системи, формулюється постановка проблеми та визначаються основні вимоги до програмного забезпечення.

У другому розділі описується операційна модель системи, варіанти використання та алгоритм роботи програмного забезпечення.

У третьому розділі розробляється архітектура системи, визначаються її апаратні та програмні компоненти, структура бази даних SQLite та її взаємодія з СУБД та Gemini API.

У четвертому розділі описується інтерфейс користувача, основні екрани вебзастосунку та результати роботи системи.

В результаті було створено робочий прототип інформаційної системи, що дозволяє користувачам вводити або сканувати штрих-коди продуктів, шукати в базі даних, відображати інформацію про продукт та отримувати аналіз складників на основі штучного інтелекту у вебінтерфейсі.

ЗМІСТ

Скорочення та умовні позначки	7
Вступ.....	8
1 Аналіз предметної області.....	11
1.1 Актуальність задачі визначення складників товарів	11
1.2 Аналіз існуючих проблем.....	12
1.3 Порівняння подібних рішень	16
1.4 Постановка задачі.....	19
2 Проєктування системи	20
2.2 Аналіз вимог до програмного забезпечення	22
2.2.1 Функціональні вимоги	24
2.2.2 Нефункціональні вимоги	26
2.3 Функціонування системи.....	28
2.3.1 Опис основних сценаріїв використання	29
2.3.2 Алгоритм роботи системи	30
3 Реалізація системи.....	34
3.1 Архітектура програмного забезпечення	34
3.2 Вибір програмного забезпечення для розробки системи.....	35
3.3 Проєктування бази даних	37
3.4 Взаємодія компонентів	39
3.5 Використання технологій ші.....	42
4 Проєктування інтерфейсу користувача.....	46
4.1 Опис інтерфейсу вебзастосунку	46

4.2 Основні екрани системи	48
4.3 Результати роботи системи	51
Висновки	53
Перелік джерел посилання	55
Додаток А Лістинги програм	57
Додаток Б Слайди презентації	70

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД	– База даних
СКБД	– Система керування базами даних
ШІ	– Штучний інтелект
API	– Application Programming Interface, програмний інтерфейс взаємодії між програмними компонентами
Frontend	– клієнтська частина програмної системи, з якою користувач взаємодіє
Backend	– серверна частина програмної системи, що виконує обробку запитів, взаємодію з базою даних та зовнішніми сервісами
UI	– User Interface, Інтерфейс користувача
HTTP	– HyperText Transfer Protocol, протокол передачі даних у мережі
JSON	– JavaScript Object Notation, текстовий формат обміну даними
UX	– User Experience, користувацький досвід
IDE	– Integrated Development Environment, інтегроване середовище розробки
SQLite	– вбудована реляційна система управління базами даних
React	– JavaScript-бібліотека для розробки інтерфейсів користувача
Node.js	– середовище виконання JavaScript на серверній стороні
Express.js	– фреймворк для створення серверних вебдодатків та API
GeminiAPI	– програмний інтерфейс доступу до моделі штучного інтелекту Gemini

ВСТУП

Сучасний розвиток інформаційних технологій суттєво впливає на повсякденне життя людей, зосереджуючись на способах збору, обробки та аналізу інформації про товари. У процесі вибору харчових продуктів споживачі часто звертають увагу не лише на ціну чи зовнішній вигляд продукту, але й на його склад, виробника, категорію, наявність потенційних алергенів, харчових добавок та інших компонентів. Це особливо актуально для людей, які страждають від харчової залежності, алергії, непереносимості певних продуктів або яким просто потрібно бути більш уважними при виборі їжі.

Одна з головних проблем полягає в тому, що інформація про склад продукту часто представлена на упаковці іншим шрифтом, що може бути важко читати або вимагати додаткових пояснень. Первинний пивовар завжди має достатньо часу або спеціальних знань для аналізу потоку інгредієнтів. Наприклад, склад може містити емульгатори, стабілізатори, ароматизатори, регулятори кислотності, добавки та інші компоненти, які не одразу очевидні. У таких умовах важливо мати розвинену інформаційну систему, яка дозволяє швидко ідентифікувати товари та отримувати вичерпну інформацію про ваш склад.

Актуальність дипломного проєкту полягає в необхідності розробки ручного програмного рішення, такого як безперервна ідентифікація товарів за штрих-кодом, збереження інформації в базі даних та використання технології штучного товарного інтелекту для пояснення складників товарів у великих кількостях для форми клієнта. Така система може бути використана в магазинах, супермаркетах, первинних лабораторіях, а також як основа для подальшої розробки мобільного або вебсервісу для персонального аналізу продукції.

Між дипломним проєктом є вебінтерфейс, який дозволяє студенту сканувати або вводити штрих-код для товару, після чого система шукає відповідний запис у базі даних та відображає основну інформацію про товар: назву, штрих-код, виробника, категорію та склад. Функція аналізу складників

додатково передана за допомогою штучного інтелекту. Після натискання кнопки відповіді склад товару передається на серверну частину, формується запит до сервісу штучного інтелекту, а на сервер передається коротка текстова викладка складників, потенційних алергенів та компонентів, що підвищують респектабельність.

Об'єктом розробки є інформаційна система для ідентифікації товарів та їх відображення на складі.

Предметом дослідження є програмне забезпечення для сканування штрих-кодів, пошуку товарів у базі даних та аналізу складників з використанням передових технологій штучного інтелекту.

Метою роботи є спрощення процесу визначення складників продукції та ідентифікації товарів за рахунок використання інтелектуальних технологій.

Структура пояснювальної записки та оформлення дипломного проекту виконані відповідно до вимог ДСТУ 3008:2015 [1], а також з урахуванням вимог до звітів у галузі науки і техніки та правил оформлення бібліографічних посилань [2].

При оформленні титульних сторінок та завдання для дипломного проекту було враховано сучасні форми Національного університету «Запорізький політехнічний інститут» [3], список літератури було підготовлено з урахуванням прикладів бібліографічного опису згідно з ДСТУ 8302:2015 [4].

Для досягнення поставленої мети необхідно виконати такі завдання:

- провести аналіз предметної області;
- визначити вимоги до розроблюваної інформаційної системи;
- спроектувати структуру системи та алгоритм її функціонування;
- обрати апаратне забезпечення системи;
- реалізувати систему з інтеграцією сервісу штучного інтелекту;
- проаналізувати отримані результати.

Практичне значення робота полягає у створенні прототипу програмного забезпечення, який демонструє принцип роботи роботизованої системи для призначення товарів на склад за штрих-кодом. Розширене рішення може бути

використане як початковий приклад для запиту клієнт-серверного вебз'єднання з базою даних та інтеграції штучного інтелекту. Крім того, систему можна розширити, додавши більшу кількість товарів, підключивши нові бази даних товарів, впровадивши спеціальний обліковий запис клієнта, скануючи історію та персональні рекомендації.

У процесі розробки використовуються такі технології: JavaScript, React, Node.js, Express.js, SQLite, сканер QR-кодів HTML5 та Gemini API. Клієнтська частина підтверджує взаємодію з клієнтом, вводячи або скануючи штрих-код та відображаючи результати. Серверна частина забезпечує обробку запитів, роботу з базою даних та передачу до сервісу штучного інтелекту. База даних використовується для зберігання інформації про товари та штучні.

Клієнтська частина веб-застосунку реалізована за допомогою бібліотеки React [5] та мови програмування JavaScript.

Серверна частина системи розроблена на платформі Node.js [6] із використанням фреймворку Express.js [7].

Для зберігання інформації про продукти використовується база даних SQLite[8].

Бібліотека html5-qrcode [9] використовується для сканування штрих-кодів.

Для аналізу складу продукту інтегровано Gemini API [10].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність задачі визначення складників товарів

У сучасних умовах споживач має доступ до великої кількості товарів, що відрізняються між собою складом, якістю, виробником, ціною, харчовою цінністю та призначенням. При цьому процес вибору товару часто посилюється тим, що інформація на упаковці може бути неповною, складною для швидкого сприйняття або незручно розташованою. Особливо це стосується складників харчових продуктів, косметичних засобів, побутової хімії та інших товарів, де наявність певних компонентів може бути важливою для здоров'я, безпеки чи особистих уподобань користувача.

Значна кількість людей має індивідуальні обмеження щодо споживання певних компонентів. До таких обмежень можна віднести алергію на молоко, сою, горіхи, глютен, непереносимість лактози, необхідність контролю споживання цукру, барвників, ароматизаторів чи інших добавок. У разі швидке отримання зрозумілої інформації склад товару як зручно, а й практично важливо. Користувачеві не досить просто побачити список інгредієнтів; часто необхідно отримати пояснення, які саме компоненти можуть бути потенційно важливими та на що слід звернути увагу.

Традиційний спосіб ознайомлення із складом товару передбачає читання інформації безпосередньо з упаковки. Однак такий підхід має низку недоліків. По-перше, текст на упаковці може бути надрукований дрібним шрифтом, по-друге, склад може містити спеціалізовані назви речовин, які не завжди зрозумілі рядовому користувачеві, по-третє, у магазині користувач має обмежений час для аналізу інформації, по-четверте, для людей з вадами зору або людей похилого віку читання складників з упаковки може бути додатково ускладненим, ну і насамперед, у зв'язку з нинішньої ситуації, багато товарів привезено з-за кордону і не завжди має переклад складників.

Вирішення цієї проблеми можливе шляхом використання інформаційних

систем, що автоматизують процес ідентифікації товару та надання інформації про нього. Одним із найпоширеніших способів ідентифікації товару є штрихкод. Штрихкод є унікальним або умовно унікальним ідентифікатором товарної позиції, що використовується у торгівлі, логістиці, складському обліку та автоматизованих системах продажу. Завдяки скануванню штрих-коду програмна система може швидко отримати код товару, виконати пошук у базі даних та надати користувачеві необхідну інформацію.

Окрему роль розвитку таких систем грають технології штучного інтелекту. На відміну від звичайного відображення складників, AI-модуль може виконувати текстовий аналіз інгредієнтів, виділяти потенційні алергени, пояснювати призначення окремих компонентів та формувати коротку рекомендацію. Це не означає, що система замінює лікаря або експерта з харчування, проте вона може бути інформаційним помічником для користувача. Саме поєднання бази даних, сканування штрих-кодів та AI-аналізу робить систему більш корисною та сучасною.

Розробка інформаційної системи визначення складників товару актуальна також із погляду навчальної та інженерної практики. В рамках цього проєкту реалізуються основні принципи клієнт-серверної архітектури, робота з базою даних, створення API, побудова інтерфейсу користувача, обробка HTTP-запитів, інтеграція зі стороннім AI-сервісом і тестування програмного забезпечення. Тема дипломного проєкту дозволяє як створити корисний програмний продукт, а й продемонструвати практичні навички розробки сучасних інформаційних систем.

1.2 Аналіз існуючих проблем

Щоб обґрунтувати доцільність розробки інформаційної системи для ідентифікації продуктів та аналізу складників, необхідно дослідити існуючі програмні рішення, які виконують аналогічні функції. Ці рішення включають

мобільні додатки для сканування штрих-кодів, відкриті бази даних харчових продуктів, системи аналізу складників продуктів, сервіси моніторингу харчування та програмні рішення для магазинів, які використовують штрих-коди для отримання інформації про продукт.

Подібні системи здебільшого орієнтовані на швидке отримання інформації про продукт після сканування штрих-коду. Деякі з цих систем спеціалізуються на харчових продуктах, інші ж – на косметичі, побутовій хімії або споживчих товарах. Основні функції цих рішень включають пошук продукту в базі даних, відображення його інвентарю, відображення інформації про харчову цінність, виявлення потенційних алергенів, надання оцінки якості або безпеки та рекомендацію альтернативних продуктів.

Одним з найвідоміших прикладів є Open Food Facts. Це відкрита база даних харчових продуктів, що містить інформацію про інгредієнти, алергени, інформацію про харчову цінність та інші дані, отримані з етикеток продуктів. Open Food Facts має API, який дозволяє отримувати інформацію про продукт за допомогою штрих-коду, зокрема після його сканування в мобільному або вебдодатку.

Перевагою Open Food Facts є його відкритість даних та можливість використовувати їх у власних програмних продуктах. Такий підхід вигідний для розробників, оскільки він усуває необхідність створювати велику базу даних продуктів з нуля, натомість використовуючи існуючі дані. Також важливо, що база даних створюється спільнотою користувачів і поступово розширюється. Однак недоліком Open Food Facts є те, що не всі продукти, особливо місцеві або менш відомі, можуть бути включені до бази даних. Крім того, якість даних залежить від точності інформації, доданої користувачами. Open Food Facts має API, який дозволяє отримувати інформацію про продукт за допомогою штрих-коду, зокрема, після його сканування в мобільному або веб-додатку [11].

Ще одним популярним рішенням є Yuka, мобільний додаток, який сканує харчові продукти та косметику, аналізує їхній склад та надає користувачам оцінку стану здоров'я. На офіційному вебсайті Yuka зазначено, що додаток розшифровує

етикетки продуктів та аналізує вплив харчових продуктів та косметики на здоров'я. Yuka також описується в магазинах додатків як програма для сканування продуктів, косметики та засобів особистої гігієни для розшифровки інгредієнтів та оцінки їхнього впливу.

Переваги Yuka включають простий інтерфейс, швидке сканування продуктів, чіткі рейтинги продуктів та рекомендації, що генеруються користувачами. Система орієнтована на пересічного споживача, який хоче швидко зрозуміти якість продукту. Недоліком цього рішення є те, що користувач не завжди бачить повну логіку процесу оцінювання та залежить від наявності продукту в базі даних додатка. Крім того, Yuka - це готовий до використання комерційний продукт, а не освітня система з відкритим кодом, тому її налаштування під конкретні вимоги користувачів або освітні проекти обмежене.

Ще один аналог EWG Healthy Living. Це мобільний додаток, який дозволяє сканувати штрих-код або шукати товар, щоб отримати оцінку косметики, харчових продуктів та засобів для чищення. Додаток надає просту оцінку за шкалою від 1 до 10, що допомагає користувачеві приймати більш обґрунтовані рішення під час покупки. На сайті EWG також зазначається, що додаток об'єднує бази даних про косметику, харчові продукти та засоби для чищення та містить оцінки для великої кількості продуктів.

Перевагою EWG Healthy Living є наявність системи оцінювання та орієнтація на безпеку продуктів. Такий підхід зручний для користувачів, які не хочуть самостійно аналізувати склад, а хочуть отримати швидкий висновок. Недоліком є те, що оцінка проводиться за методикою конкретної організації, і користувач не завжди може адаптувати критерії оцінки до власних потреб. Крім того, система більше орієнтована на ринок США та продукти, представлені у відповідних базах даних.

До подібних рішень також можна віднести додатки для моніторингу харчування, наприклад, MyFitnessPal. Основне призначення таких систем полягає не стільки в аналізі складників продукту, скільки у веденні харчового щоденника, підрахунку калорій, білків, жирів та вуглеводів. При цьому MyFitnessPal має

функцію сканування штрих-кодів для додавання продуктів до харчового щоденника, а база даних продуктів використовується для отримання інформації про харчову цінність.

Перевагою таких систем є велика база даних продуктів та зручність для користувачів, які контролюють свій раціон. Недоліком є те, що основний акцент робиться на калоріях та поживних речовинах, а не на поясненні складників продукту. Тобто користувач може отримати інформацію про енергетичну цінність, але не завжди отримує просте пояснення потенційних алергенів чи окремих інгредієнтів.

Окрім спеціалізованих додатків, існують також внутрішні системи магазинів та супермаркетів. Вони використовують штрих-коди для обліку товарів, продажів через касові апарати, перевірки цін, обліку запасів та логістики. У деяких випадках мобільні додатки магазинів дозволяють сканувати продукт та переглядати інформацію про нього. Однак такі системи зазвичай обмежені продуктами певної торгової мережі та не завжди надають детальний аналіз запасів.

Таким чином, аналіз існуючих рішень показує, що на ринку вже існують системи, які дозволяють сканувати штрих-код та отримувати інформацію про товар. Однак більшість із них мають певні обмеження: залежність від закритої бази даних, орієнтація на певний ринок, обмежені можливості адаптації, відсутність власної локальної бази даних або недостатня гнучкість у поясненні стану товарних запасів. Саме тому розробка навчального прототипу інформаційної системи, що поєднує сканування штрих-коду, локальну базу даних та AI-аналіз стану товарних запасів, є доцільною та практично виправданою.

1.3 Порівняння подібних рішень

Для більш чіткого порівняння подібних систем доцільно виділити основні критерії, які є важливими для розробленого дипломного проєкту. До таких критеріїв належать: можливість сканування штрих-коду, наявність бази даних продуктів, відображення складників, аналіз інгредієнтів, використання штучного інтелекту або алгоритмів оцінки, доступність для користувача, можливість адаптації та придатність для використання в освітніх цілях.

Таблиця 1.1 – Порівняння аналогічних систем

Система	Призначення	Переваги	Недоліки
OpenFoodFacts	Відкрита база даних продуктів харчування та API для отримання даних штрих-кодів	Відкритість, доступність активного фармацевтичного інгредієнта (API), можливість отримання інформації про склад та харчову цінність	Не всі локальні товари є в базі, якість даних залежить від користувачів
Yuka	Сканування харчових продуктів і косметики, оцінювання впливу на здоров'я	Зручний інтерфейс, швидкий аналіз, рейтинг товару	Закрита логіка оцінювання, залежність від бази застосунку

Продовження таблиці 1.1

Система	Призначення	Переваги	Недоліки
EWG Healthy Living	Оцінка косметики, харчових продуктів та засобів для чищення	Система оцінювання, зосереджена на безпеці продукції	Зосередження переважно на певному ринку з обмеженою адаптацією до власних критеріїв.
MyFitnessPal	Контроль харчування, підрахунок калорій та поживних речовин	Велика БД продуктів, сканування штрих-кодів, зручно для щоденника	Основна увага приділяється калоріям, а не поясненню складників
Внутрішні системи магазинів	Облік, продажі та перевірка інформації про продукцію	Інтеграція з роздрібною мережею, актуальні ціни та залишки	Обмеження товарами з певного магазину, відсутність універсального аналізу складників
Розроблювана система	Ідентифікація продукту за штрих-кодом, відображення товарних запасів та аналіз ШІ	Локальна база, простота, адаптивність, інтеграція штучного інтелекту, навчальна цінність	Прототип має обмежену кількість товарів у базі, потребує подальшого розширення

Порівняння показує, що кожна з розглянутих систем має своє специфічне застосування. Open Food Facts зручне джерело відкритих даних, Yuka та EWG Healthy Living зосереджені на оцінці продуктів, MyFitnessPal використовується

переважно для моніторингу харчування, а системи для роботи в магазинах призначені для внутрішніх бізнес-процесів. Однак система, що розробляється, має інше призначення: вона створюється як навчальний та демонстраційний прототип, що демонструє повний робочий процес клієнт-серверного застосунку від введення або сканування штрих-коду до отримання даних з бази даних та аналізу інгредієнтів за допомогою штучного інтелекту.

Відмінною особливістю системи, що розробляється, є реалізація локальної бази даних. Це дозволяє контролювати структуру збережених даних, самостійно додавати продукти та модифікувати склад, виробника, категорію та іншу інформацію. Такий підхід підходить для дипломного проєкту, оскільки демонструє проєктування користувацької бази даних та взаємодію програмного забезпечення з СУБД. Систему можна додатково розширити, підключивши зовнішній API, такий як Open Food Facts, але в рамках базової реалізації локальна база даних є більш керованою та зрозумілою.

Ще однією важливою відмінністю є використання модуля штучного інтелекту для отримання інформації про інгредієнти. У багатьох подібних системах користувач отримує оцінку або список інгредієнтів, але не завжди має доступ до пояснення у відкритому текстовому вигляді. У системі, що розробляється, штучний інтелект використовується як допоміжний інструмент для формування коротких висновків щодо складників продукту, потенційних алергенів та компонентів, на які слід звернути увагу. Це підвищує інформативність системи та робить її зручнішою для користувача.

Важливо також зазначити, що система, що розробляється, не претендує на повну заміну медичних порад чи професійного аналізу продукту. Її основна мета - надання довідкової інформації та демонстрація можливостей сучасних вебтехнологій та штучного інтелекту. Тому доцільно використовувати нейтральну мову у відповідях модуля штучного інтелекту, уникати категоричних висновків та вказувати, що рекомендації надаються лише в інформаційних цілях.

На основі порівняння можна зробити висновок, що розробка індивідуальної інформаційної системи є доцільною, оскільки вона дозволяє реалізувати саме ті

функції, які необхідні для дипломного проєкту: сканування або введення штрих-коду, пошук продукту в базі даних, відображення складників, аналіз складників на основі штучного інтелекту та демонстрацію апаратних та програмних компонентів системи.

1.4 Постановка задачі

На основі аналізу предметної області та подібних систем можна сформулювати основне завдання дипломного проєкту. Необхідно розробити інформаційну систему, яка забезпечує ідентифікацію продукту за штрих-кодом, отримання інформації про продукт з бази даних та додатковий аналіз складників за допомогою технологій штучного інтелекту.

Розроблена система має бути реалізована у вигляді вебдодатку. Такий підхід був обраний з кількох причин. По-перше, вебдодаток не вимагає встановлення окремої мобільної програми на пристрій користувача. По-друге, вебтехнології дозволяють запускати систему на різних пристроях, зокрема на персональному комп'ютері, ноутбучі, планшеті чи смартфоні. По-третє, браузер може використовувати камеру пристрою для сканування штрих-коду, що дозволяє використовувати камеру як апаратний сканер.

В рамках дипломного проєкту необхідно реалізувати такі основні функції системи:

- ручне введення штрих-коду товару;
- можливість сканування штрих-коду за допомогою камери пристрою;
- передача отриманого штрих-коду з клієнтської частини на сервер;
- пошук товару в базі даних за значенням штрих-коду;
- відображення інформації про знайдений товар;
- відображення назви товару, категорії, виробника, штрих-коду та складників;

- обробка ситуації, коли товар не знайдено в базі даних;
- відправлення складників товару на сервер для аналізу за допомогою ШІ;
- отримання текстового виводу від сервісу ШІ;
- відображення результату аналізу за допомогою ШІ в інтерфейсі користувача.

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Загальна структура системи

Вебзастосунок, що розробляється, призначений для автоматизації процесу отримання інформації про склад продукту. Основним користувачем системи є звичайний споживач, який бажає швидко дізнатися про склад продукту і отримати короткий аналіз його компонентів. Система також може бути корисною як навчальний приклад створення клієнт-серверного вебдодатка з використанням бази даних, API та штучного інтелекту.

Система має працювати за таким загальним принципом. Користувач відкриває вебпрограму в браузері, потім вводить штрих-код продукту вручну або запускає сканер. Якщо використовується сканер, камера пристрою зчитує штрих-код, а спеціальна бібліотека розпізнає його числове значення. Отриманий штрих-код передається на серверну частину програми. Сервер виконує запит до бази даних SQLite та перевіряє, чи існує продукт із таким штрихкодом. Якщо продукт знайдено, система повертає його назву, категорію, виробника та склад. Якщо продукт не знайдено, користувач отримує відповідне повідомлення.

Після отримання інформації про продукт, користувач може використовувати функцію аналізу за допомогою ШІ. Для цього склад товару відправляється на бекенд, де формується запит до API Gemini. Сервіс штучного інтелекту опрацьовує текстовий опис складників та повертає відповідь

українською мовою. Результат відображається у спеціальному блоці інтерфейсу користувача.

Система, що розробляється, складається з таких основних компонентів:

- клієнтський компонент, відповідальний за інтерфейс користувача;
- модуль сканування штрих-кодів за допомогою камери пристрою;
- серверний компонент, який отримує запити від клієнта;
- база даних, що зберігає товари;
- API для пошуку товарів за штрих-кодом;
- модуль для взаємодії з Gemini API;
- інтерфейс для відображення результатів пошуку та аналізу за допомогою штучного інтелекту (рисунок 1.1).

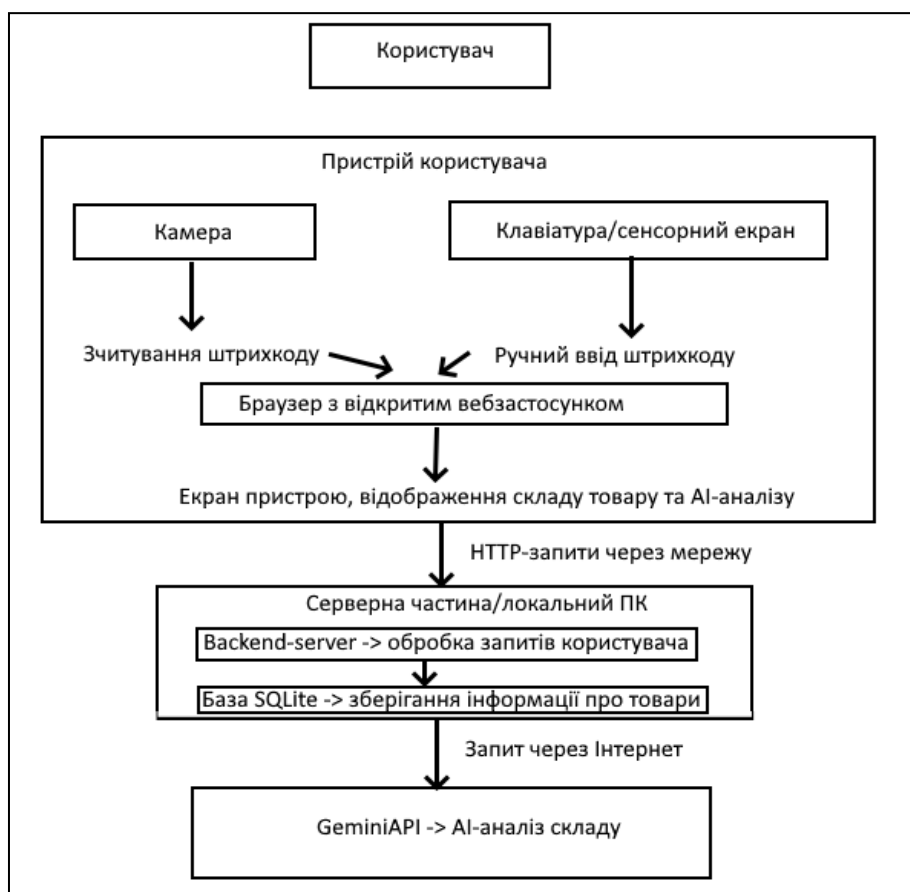


Рисунок 1.1 – Загальна структура системи

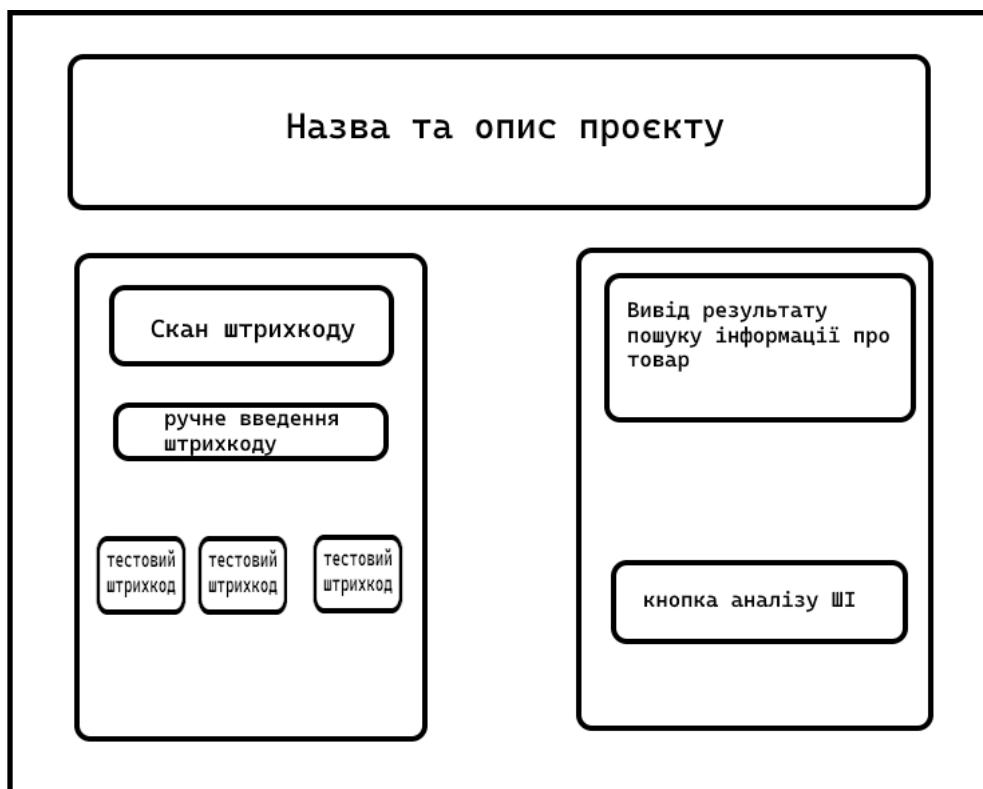


Рисунок 1.2 - Структура інтерфейсу системи

Для цього дипломного проєкту система реалізується як прототип. Це означає, що база даних містить обмежену кількість тестових товарів, але структура програмного забезпечення дозволяє подальше розширення, включаючи додавання більшої кількості товарів, підключення зовнішніх баз даних та реалізацію персональних профілів користувачів або історії сканування.

2.2 Аналіз вимог до програмного забезпечення

Вимоги до програмного забезпечення визначають, які функції повинна виконувати система та яким характеристикам вона повинна відповідати. Для розробленої системи вимоги можна розділити на функціональні та нефункціональні.

Функціональні вимоги описують конкретні можливості системи. Вони визначають, які дії може виконувати користувач, як система повинна обробляти дані та які результати слід отримувати. Для цього проєкту основними функціональними вимогами є введення штрих-коду, сканування штрих-коду, пошук товару в базі даних, відображення інформації про товар, обробка помилок, аналіз запасів за допомогою ІІІ та повернення результату користувачеві.

Нефункціональні вимоги описують якісні характеристики системи. До них належать вимоги до зручності використання, швидкості, надійності, безпеки, доступності та можливості подальшого розширення. Для розробленого вебзастосунку важливо, щоб інтерфейс був простим, а користувач міг швидко виконати основну дію - знайти товар за штрих-кодом. Також важливо, щоб ключ API до сервісу ІІІ не був доступний у клієнтському коді, оскільки це може призвести до його несанкціонованого використання.

Основні вимоги до програмного забезпечення наведено в таблиці 1.2.

Таблиця 1.2 - Основні вимоги до програмного забезпечення

Категорія вимоги	Опис вимоги
Введення даних	Система повинна дозволяти користувачу вводити штрихкод вручну
Сканування	Система повинна давати сканувати штрихкод за допомогою камери пристрою
Пошук товару	Система повинна виконувати пошук в базі даних
Відображення товару	Система повинна показувати назву, виробника, штрихкод, склад товару
Обробка відсутніх даних	Якщо товар не знайдено, система відображає повідомлення про відсутність товару у базі

Продовження таблиці 1.2

Категорія вимоги	Опис вимоги
Серверна обробка	Усі запити до бази даних та AI-сервісу повинні виконуватися на серверній частині
Безпека	API-ключ Gemini не повинен зберігатися у frontend-частині
Зручність	Інтерфейс користувача повинен бути простим, зрозумілим та адаптованим до демонстрації
Розширюваність	Система повинна мати можливість подальшого додавання нових товарів та функцій

Аналіз вимог показує, що система повинна виконувати не лише базову функцію пошуку товару, а й забезпечувати додаткову обробку даних. Саме AI-аналіз складників відрізняє розроблювану систему від простого довідника товарів. Завдяки цьому користувач отримує не тільки інформацію з бази даних, а й текстове пояснення, сформоване на основі складників товару. Під час визначення вимог до програмного забезпечення враховувалися загальні підходи до розробки вимог, описані у стандарті ISO/IEC/IEEE 29148:2018 [12].

2.2.1 Функціональні вимоги

Функціональні вимоги визначають конкретні дії, які повинна виконувати інформаційна система. Для цього дипломного проєкту функціональні вимоги пов'язані з основними сценаріями використання системи: введення штрих-коду, сканування товару, пошук у базі даних, перегляд інформації про товар та запуск ШІ-аналізу.

Ця функція необхідна, оскільки не завжди зручно використовувати камеру під час тестування або демонстрації системи. Крім того, ручне введення дозволяє

перевірити роботу системи, навіть якщо камера пристрою недоступна або браузер не надав дозволу на її використання.

Для цього в клієнтській частині використовується бібліотека, яка отримує відеопотік з камери браузера, аналізує зображення та визначає значення штрих-коду. Після успішного розпізнавання штрих-код автоматично передається на сервер для подальшої обробки.

Серверна частина повинна прийняти штрих-код, виконати SQL-запит до бази даних SQLite та повернути знайдений запис. Якщо товар знайдено, відповідь повинна містити основні дані про товар. Якщо запис відсутній, сервер повинен повернути повідомлення про помилку з відповідним статусом.

Після отримання відповіді від сервера, фронтенд-частина повинна показувати користувачеві назву товару, категорію, виробника, штрих-код та склад. Візуально ця інформація повинна бути розміщена в чіткому блоці результатів.

Після відображення товару користувач може натиснути кнопку аналізу складників. Після цього фронтенд надсилає назву товару та його склад на сервер. Бекенд формує текстовий запит до Gemini API та отримує відповідь. Результат аналізу за допомогою штучного інтелекту повертається клієнту та відображається у спеціальному блоці.

Шостою функціональною вимогою є обробка помилок. Система має коректно реагувати на такі ситуації:

- користувач не ввів штрихкод;
- Товар відсутній у базі даних;
- сервер недоступний;
- AI-сервіс не повернув відповіді;
- API-ключ не налаштований;
- сталася помилка з'єднання.

Таблиця 2.1 - Основні функціональні вимоги

Функція	Опис
Ручне введення штрихкоду	Вводиться штрихкод у текстове поле та натискається кнопку пошуку
Сканування штрихкоду	Користувач запускає сканер, камера зчитує штрихкод товару
Передача штрихкоду на сервер	Frontend надсилає HTTP-запит на backend
Пошук у базі даних	Backend виконує пошук товару в SQLite за значенням штрихкоду
Виведення результату	Система показує товар і його склад
Повідомлення про помилку	Якщо товар не знайдено або сервер недоступний, система показує повідомлення
AI-аналіз складників	Система надсилає склад товару до AI-сервісу та отримує текстове пояснення
Відображення AI-висновку	Результат аналізу показується у спеціальному блоці інтерфейсу

Таким чином, функціональні вимоги охоплюють повний цикл роботи системи: від отримання штрих-коду до генерації результату для користувача. Реалізація цих вимог забезпечує працездатність основної логіки програмного продукту.

2.2.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якість системи, її зручність, надійність, безпеку та можливість подальшого розвитку. Для інформаційної системи ідентифікації товарів та аналізу їх складників ці вимоги є важливими, оскільки система має бути не тільки функціональною, але й зручною у використанні.

Користувач повинен швидко зрозуміти, як користуватися системою. Основні елементи інтерфейсу мають бути помітними: кнопка запуску сканера, поле введення штрих-коду, кнопка пошуку, блок результатів та кнопка ШІ-аналізу. Текстові повідомлення мають бути зрозумілими та написаними українською мовою.

Пошук товару в локальній базі даних має виконуватися швидко, оскільки база даних SQLite зберігається локально на сервері. Запит до ШІ-сервісу може зайняти більше часу, оскільки залежить від зовнішнього API та швидкості мережевого з'єднання. Тому під час ШІ-аналізу система повинна показувати користувачеві, що запит обробляється.

Система повинна коректно обробляти типові помилки. Наприклад, якщо користувач ввів штрих-код, якого немає в базі даних, програма не повинна завершуватися неправильно. Натомість має відображатися повідомлення про те, що товар не знайдено. Аналогічно, якщо сервіс ШІ недоступний, користувач повинен отримати чітке повідомлення про помилку.

Оскільки система використовує зовнішній сервіс ШІ, ключ API не повинен зберігатися у відкритому коді клієнта. Він має бути розміщений у файлі середовища .env на стороні сервера. Фронтенд повинен отримувати доступ лише до власного API бекенду, а сам бекенд робить запит до API Gemini. Це зменшує ризик витоку ключа.

У майбутньому система повинна мати можливість додавати нові товари, розширювати структуру бази даних, підключати зовнішні джерела інформації, реалізовувати історію сканування або персональні рекомендації. Обрана клієнт-серверна архітектура дозволяє поступово розширювати функціональність без повного перероблення системи.

Оскільки система реалізована як вебдодаток, вона повинна працювати в сучасних браузерях. Базова логіка пошуку товарів доступна як на комп'ютері, так і на мобільному пристрої. Сканування за допомогою камери може залежати від налаштувань браузера, дозволів користувача та вимог безпечного з'єднання.

Дотримання нефункціональних вимог дозволяє зробити систему більш якісною, зручною та придатною для подальшого розвитку.

2.3 Функціонування системи

Загальна модель роботи вебзастосунок відображає послідовність взаємодії між користувачем, клієнтською частиною, серверною частиною, базою даних та сервісом штучного інтелекту. Ця модель дозволяє зрозуміти, які етапи проходить запит користувача від моменту введення або сканування штрих-коду до отримання результату на екрані.

Робота системи починається з відкриття користувачем вебпрограми. Клієнтська частина системи реалізована у вигляді вебінтерфейсу, який містить поле для ручного введення штрих-коду, кнопку запуску сканера, кнопки для демонстрації роботи системи та блок для відображення результатів. Користувач може самостійно ввести штрих-код продукту або використовувати камеру пристрою для його сканування.

Після отримання штрих-коду фронтенд-частина генерує HTTP-запит до бекенд-частини. Сервер отримує штрих-код як параметр запиту і шукає відповідний продукт у базі даних SQLite. Якщо запис із таким штрих-кодом існує, база даних повертає інформацію про продукт. Бекенд генерує відповідь у форматі JSON і передає її клієнтській частині. Після цього фронтенд відображає дані у зручному для користувача вигляді.

Якщо користувач хоче отримати додаткове пояснення складників, він натискає кнопку "Аналіз за допомогою ШІ". У цьому випадку фронтенд відправляє назву продукту та його склад на бекенд. Серверна частина створює текстовий запит до API Gemini. Сервіс ШІ аналізує склад продукту та повертає відповідь українською мовою. Результат передається назад у вебінтерфейс та відображається у спеціальному блоці аналізу за допомогою ШІ.

Загальну структуру системи можна описати так:

Користувач → Вебінтерфейс React → Введення або сканування штрихкоду → HTTP-запит на сервер Node.js / Express → Пошук товару в базі даних SQLite → Повернення інформації про товар → AI-аналіз складників через Gemini API → Відображення результату користувачу

2.3.1 Опис основних сценаріїв використання

Для повного розуміння роботи інформаційної системи необхідно описати основні сценарії її використання. Сценарій використання показує, які дії виконує користувач і як система реагує на ці дії. У розробленому вебдодатку можна виділити кілька основних сценаріїв: пошук товару за введеним штрих-кодом, сканування штрих-коду камерою, перегляд інформації про товар, аналіз запасів за допомогою штучного інтелекту та обробка ситуації, коли товар не знайдено.

Перший сценарій передбачає ручне введення штрих-коду. Ця опція важлива для тестування та демонстрації системи, оскільки дозволяє перевірити роботу логіки програми навіть без використання камери. Користувач вводить штрих-код у текстове поле та натискає кнопку пошуку. Після цього система надсилає запит на сервер, сервер шукає товар у базі даних та повертає результат.

Другий сценарій пов'язаний зі скануванням штрих-коду за допомогою камери. Користувач натискає кнопку запуску сканера, після чого браузер може запросити дозвіл на використання камери. Після надання дозволу камера активується, і спеціальна бібліотека виконує розпізнавання штрих-коду. Після успішного розпізнавання отриманий код автоматично використовується для пошуку товару в базі даних.

Третій сценарій передбачає перегляд інформації про товар. Якщо товар знайдено, система відображає його основні дані: назву, штрих-код, виробника, категорію та склад. Цей етап є основним результатом роботи системи, оскільки користувач отримує необхідну інформацію про товар у зрозумілій формі.

Четвертий сценарій пов'язаний з AI-аналізом складників. Після перегляду інформації про товар користувач може натиснути кнопку аналізу складників за допомогою штучного інтелекту. Система передає склад товару на сервер, сервер

зв'язується з Gemini API, отримує відповідь та повертає її клієнту. В результаті користувач бачить короткий текстовий аналіз складників, потенційних алергенів та загальну рекомендацію.

П'ятий сценарій описує ситуацію, коли товар не знайдено. Якщо користувач вводить штрих-код, якого немає в базі даних, система не повинна завершити роботу з помилкою. Вона повинна відображати чітке повідомлення про те, що товару з таким штрих-кодом немає в базі даних. Це дозволяє користувачеві спробувати інший штрих-код або зрозуміти, що базу даних потрібно розширити.

Описані сценарії охоплюють основні варіанти роботи користувача із системою. Вони демонструють, що система має передбачувану логіку роботи, підтримує як успішні, так і помилкові ситуації, а також забезпечує взаємодію з базою даних і сервісом штучного інтелекту.

2.3.2 Алгоритм роботи системи

Алгоритм програмної частини визначає послідовність дій, які система виконує під час обробки запиту користувача. У розробленій системі основним вхідним параметром є штрих-код продукту. Саме це значення використовується для пошуку в базі даних. Після успішного пошуку користувач може додатково розпочати аналіз складників за допомогою штучного інтелекту. При побудові алгоритму системи було враховано загальні підходи до прийняття рішень та послідовної обробки даних у програмних системах [13].

Все починається з відкриття вебзастосунку. На головній сторінці користувач бачить елементи керування: кнопку запуску сканера, поле для ручного введення штрих-коду, кнопку пошуку та блок результатів. Далі користувач обирає спосіб отримання штрих-коду: ручне введення або сканування.

Якщо користувач вводить штрих-код вручну, система перевіряє, чи поле порожнє. Якщо поле порожнє, користувачеві відображається повідомлення про необхідність введення штрих-коду. Якщо штрих-код введено, фронтенд надсилає запит до бекенд-частини.

Якщо користувач використовує сканер, система активує камеру пристрою. Після розпізнавання штрих-коду отримане значення автоматично передається до

функції пошуку продукту. Подальша логіка роботи однакова для обох способів введення.

Бекенд отримує штрих-код і виконує SQL-запит до таблиці товарів у базі даних SQLite. Якщо товар знайдено, сервер повертає інформацію про нього. Якщо товар відсутній, сервер повертає повідомлення про помилку. Фронтенд обробляє відповідь і або відображає дані про товар, або відображає повідомлення про те, що товар не знайдено.

Після успішного знаходження товару користувач може виконати аналіз запасів за допомогою штучного інтелекту. Для цього фронтенд надсилає назву товару та його наявність на складі на сервер. Бекенд перевіряє наявність на складі та ключ Gemini API. Якщо всі умови виконано, сервер генерує текстовий запит до Gemini API. Отриманий результат повертається на фронтенд і відображається користувачеві (рисунок 2.1).

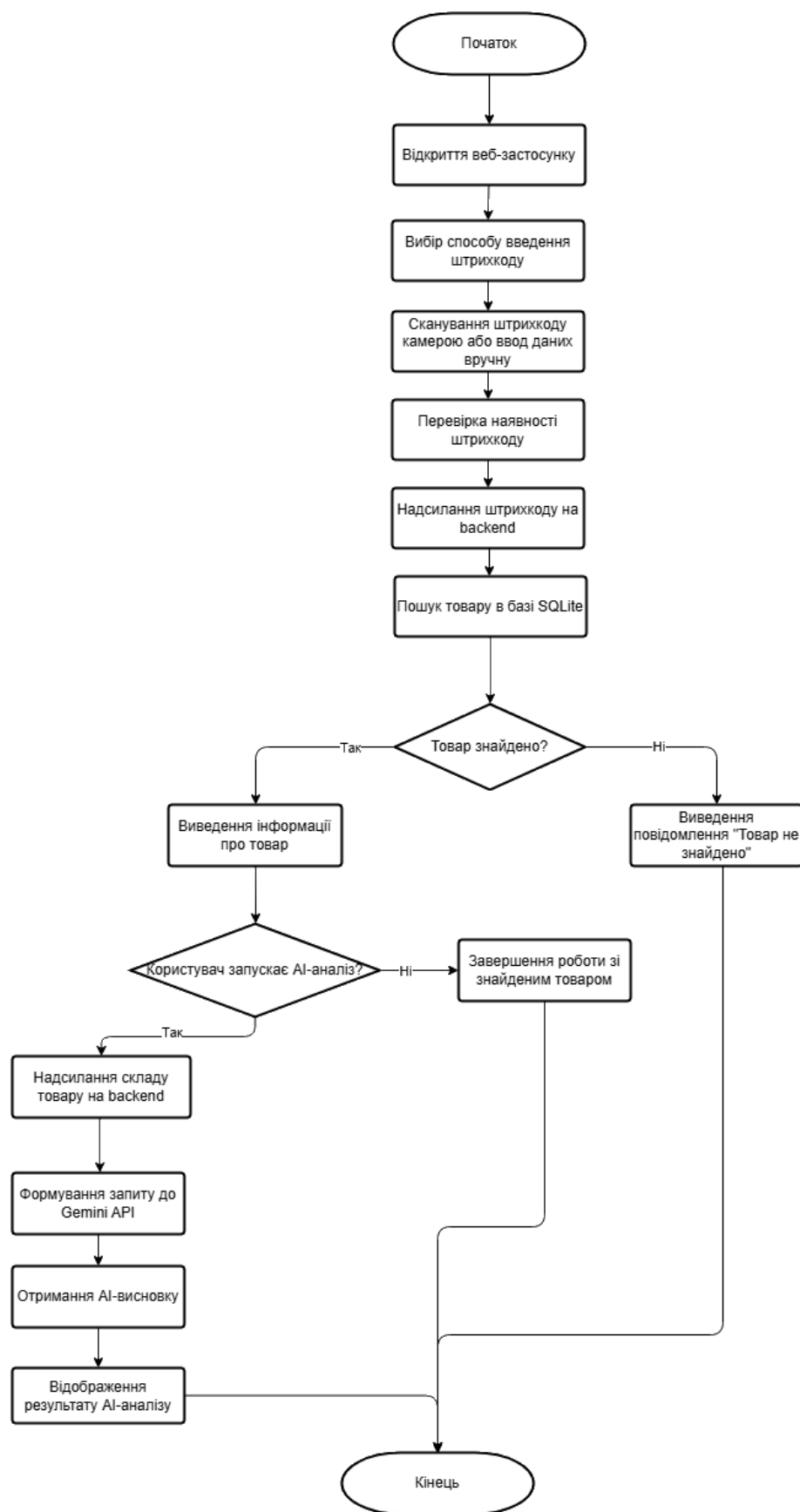


Рисунок 2.1 – Блок-схема алгоритму функціонування вебзастосунку

2.4 Вибір апаратного забезпечення вебзастосунку

Апаратне забезпечення системи визначає фізичні пристрої, які користувач використовує для взаємодії з програмним забезпеченням, та спосіб отримання вихідних даних. У цьому дипломному проєкті не використовується спеціалізований сканер штрих-кодів. Цю роль виконує камера пристрою користувача.

Такий підхід має кілька переваг. По-перше, камера вбудована в більшість сучасних ноутбуків, планшетів та смартфонів, що усуває потребу в додатковому обладнанні. По-друге, використання камери робить систему доступною для широкого кола користувачів. По-третє, вебдодаток можна запускати через браузер, що зменшує потребу в установці додаткового програмного забезпечення.

Сканер у цій системі є апаратно-програмним модулем, що складається з камери пристрою та бібліотеки розпізнавання штрих-кодів. Камера фіксує зображення штрих-коду, а програмна бібліотека аналізує відеопотік та визначає числове значення коду. Після успішного розпізнавання штрих-код передається на клієнтську частину вебдодатку, а потім надсилається на сервер.

Обробка даних у системі виконується на програмному рівні, а не на рівні камери. Камера фіксує лише зображення. Розпізнавання штрих-кодів виконується в браузері за допомогою спеціальної бібліотеки, тоді як пошук товарів та аналіз за допомогою штучного інтелекту виконуються на стороні сервера. Це дозволяє розділити процес на кілька етапів: зчитування, розпізнавання, передача, пошук та аналіз.

Результати обробки даних надсилаються назад до інтерфейсу користувача. Користувач бачить інформацію на екрані пристрою: назву продукту, штрих-код, виробника, категорію, склад та результат аналізу за допомогою штучного інтелекту.

Апаратну складову системи можна описати наступним чином:

- камера пристрою – для захоплення зображення штрих-коду;

- екран пристрою – для відображення інформації про продукт;
- клавіатура або сенсорний екран – для ручного введення штрих-коду та керування системою;
- пристрій користувача – для запуску вебзастосунку у браузері;
- сервер або локальний комп'ютер – для виконання серверної частини та роботи з базою даних.

3 РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Архітектура програмного забезпечення

Загальна архітектура розробленої системи заснована на розподілі програмного забезпечення на фронтенд та бекенд. Такий підхід типовий для сучасних вебдодатків, оскільки дозволяє розробляти інтерфейс користувача і серверну логіку окремо, також враховувалися загальні підходи до моделювання програмних систем, що використовуються в об'єктно-орієнтованому проектуванні [14].

Фронтенд запускається у браузері користувача. Він відповідає за відображення сторінки вебпрограми, роботу з полем введення штрих-коду, запуск сканера, обробку дій користувача та відображення результатів. Саме з фронтендом користувач взаємодіє безпосередньо.

Бекенд працює як серверний додаток. Він приймає HTTP-запити від фронтенду, шукає продукт у базі даних, повертає інформацію про продукт у форматі JSON та звертається до API Gemini для аналізу запасів. Бекенд також виступає як проміжний шар між інтерфейсом користувача, базою даних і зовнішнім сервісом штучного інтелекту.

Для зберігання інформації про продукти використовується база даних SQLite. У базі даних зберігаються такі дані: штрих-код, назва продукту,

інвентаризація, виробник та категорія. Пошук продукту здійснюється за штрих-кодом, оскільки це основний ідентифікатор продукту у системі.

Для аналізу складників продукту використовують зовнішній сервіс Gemini API. Після того, як користувач знайде продукт у базі даних, він може запустити аналіз за допомогою ШІ. У цьому випадку бекенд передає назву продукту та його склад Gemini API, отримує текстовий результат і повертає його на фронтенд.

Важливою особливістю такої архітектури є те, що frontend не має прямого доступу до бази даних і Gemini API. Це дозволяє забезпечити кращий рівень безпеки, оскільки ключ доступу до AI-сервісу зберігається тільки на серверній частині. Крім того, така архітектура спрощує подальше розширення системи, оскільки до backend-частини можна додавати нові маршрути, нові таблиці бази даних або інтеграцію з іншими зовнішніми сервісами.

3.2 Вибір програмного забезпечення для розробки системи

Програмна складова системи включає всі програмні компоненти, що забезпечують роботу вебзастосунку. Вона складається з клієнтської частини, серверної частини, бази даних та модуля інтеграції з сервісом штучного інтелекту.

Клієнтська частина реалізована за допомогою JavaScript та бібліотеки React. React використовується для побудови користувацького інтерфейсу, відображення компонентів сторінки, обробки станів застосунку та взаємодії з користувачем. Клієнтська частина реалізує поле введення штрих-коду, тестування кнопок товару, запуск сканера, блок результатів та блок аналізу штучного інтелекту.

Серверна частина реалізована на Node.js з використанням фреймворку Express.js. Node.js дозволяє виконувати JavaScript на стороні сервера, а Express.js використовується для створення API-маршрутів. Сервер обробляє запити від фронтенду, шукає товари в базі даних, додає тестові товари та звертається до

Gemini API для аналізу запасів. Приклад підключення основних модулів наведено в лістингу 3.1.

Лістинг 3.1 - Підключення основних бібліотек серверної частини

```
require("dotenv").config();
const express = require("express");
const cors = require("cors");
const Database = require("better-sqlite3");
const { GoogleGenAI } = require("@google/genai");
const app = express();
const PORT = 5000;
const ai = new GoogleGenAI({
  apiKey: process.env.GEMINI_API_KEY,
});
app.use(cors());
app.use(express.json());
```

Наведений вище фрагмент містить файл середовища .env, який зберігає ключ доступу до API Gemini. Бібліотека express використовується для створення бекенду та маршрутів API. Бібліотека cors забезпечує обмін даними між клієнтом та бекендом. Модуль better-sqlite3 використовується для роботи з базою даних SQLite, а GoogleGenAI використовується для взаємодії зі службою штучного інтелекту Gemini.

Для зберігання даних використовується SQLite. Це вбудована реляційна база даних, яка зберігається як окремий файл. Для прототипу диплома SQLite є зручним рішенням, оскільки не вимагає налаштування окремого сервера баз даних та дозволяє швидко створити робочу систему. А для взаємодії серверної частини з самою базою даних SQLite використовувалася бібліотека better-sqlite3, яка дозволяє виконувати SQL-запити безпосередньо з коду Node.js [15].

Для взаємодії з базою даних у бекенді використовується бібліотека better-sqlite3. Вона дозволяє створювати таблиці, виконувати SQL-запити, додавати

записи та отримувати дані з бази даних. Система використовує таблицю `products`, яка зберігає інформацію про товари.

Для сканування штрих-кодів у браузері використовується бібліотека `html5-qrcode`.

`Gemini API` використовується для інтеграції зі штучним інтелектом. Бекенд генерує текстовий запит, що містить назву товару та його запаси, а потім надсилає його до служби штучного інтелекту. У відповідь система отримує текстовий аналіз, який повертається до фронтенду та відображається користувачеві.

3.3 Проєктування бази даних

База даних є важливою частиною розробленої системи, оскільки саме в ній зберігається інформація про товари. У дипломному проєкті використовується база даних `SQLite`. Її достатньо для реалізації прототипу, оскільки вона дозволяє зберігати дані локально та швидко здійснювати пошук за штрих-кодом, а також не потребує окремого серверного процесу та підходить для створення навчального прототипу.

Основною таблицею бази даних є таблиця `products`. Вона призначена для зберігання інформації про товари, які може знайти система. Кожен запис у таблиці відповідає одному товару(див. таблиця 3.1).

Таблиця 3.1 - Структура таблиці `products`

Назва поля	Тип даних	Призначення
Id	INTEGER	Унікальний ID-запису
Barcode	TEXT	Штрихкод товару
Name	TEXT	Назва товару
Composition	TEXT	Склад товару
manufacturer	TEXT	Виробник товару

Продовження таблиці 3.1

Назва поля	Тип даних	Призначення
category	TEXT	Категорія товару

Поле `id` використовується як первинний ключ таблиці. Воно автоматично збільшується при додаванні нового товару. Поле `barcode` зберігає штрих-код товару і використовується для пошуку. Поле `name` містить назву товару, поле `composition` - його склад, поле `manufacturer` - інформацію про виробника, а поле `category` - категорію товару.

Для створення таблиці використовується наступний SQL-запит (лістинг 3.2):

Лістинг 3.2 – приклад SQL-запиту

```
CREATE TABLE IF NOT EXISTS products (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    barcode TEXT UNIQUE NOT NULL,
    name TEXT NOT NULL,
    composition TEXT NOT NULL,
    manufacturer TEXT,
    category TEXT
);
```

Цей фрагмент відкриває або створює базу даних `products.db`. Команда `CREATE TABLE IF NOT EXISTS` створює таблицю `products`, лише якщо вона ще не існує. Поле `id` використовується як первинний ключ, `barcode` зберігає штрих-код продукту, `name` - назву продукту, `composition` - склад, `manufacturer` - виробника, а `category` - категорію продукту. Поле `barcode` використовується для подальшого пошуку продукту.

Важливою особливістю таблиці є те, що поле штрих-коду має бути унікальним. Це необхідно для того, щоб один штрих-код відповідав лише одному

продукту в базі даних. Якщо користувач вводить або сканує штрих-код, система шукає це поле.

База даних забезпечує зберігання тестових продуктів, які використовуються для демонстрації системи. Наприклад, до бази даних можна додати такі продукти: молоко, шоколад, яблучний сік, полуничний йогурт. Кожен з них має свій власний штрих-код, назву, виробника, категорію та склад.

Клієнтська частина також містить функцію, яка відповідає за надсилання штрих-коду на серверну частину та обробку відповіді. Цей фрагмент коду показує зв'язок між інтерфейсом користувача та API пошуку товарів.

3.4 Взаємодія компонентів

Взаємодія програмного забезпечення з СУБД відбувається на стороні сервера системи. Фронтенд не має прямого доступу до бази даних. Це означає, що всі операції з даними виконуються через API бекенду. Такий підхід є більш безпечним і дозволяє централізовано контролювати доступ до інформації.

Після того, як користувач вводить або сканує штрих-код, фронтенд надсилає HTTP-запит до бекенду. Маршрут, який використовується для пошуку продукту: `GET /api/products/:barcode`.

Основна взаємодія між клієнтом, сервером та базою даних відбувається під час пошуку товару за штрих-кодом. Для цього на сервері реалізовано API-маршрут, який приймає штрих-код як параметр запиту та виконує пошук у таблиці `products`. Приклад такого маршруту наведено в лістингу 3.3.

Лістинг 3.3 – Пошук товару за штрих-кодом

```
app.get("/api/products/:barcode", (req, res) => {
  const { barcode } = req.params;

  const product = db
```

```

        .prepare("SELECT * FROM products WHERE barcode = ?")
        .get(barcode);
    if (!product) {
        return res.status(404).json({
            message: "Товар з таким штрихкодом не знайдено",
        });
    }
    res.json(product);
});

```

У наведеному вище фрагменті сервер отримує значення штрих-коду з параметрів HTTP-запиту. Далі виконується параметризований SQL-запит до таблиці products. Якщо продукт знайдено, сервер повертає його дані у форматі JSON. Якщо запис відсутній, клієнтська сторона отримує повідомлення про те, що продукт не знайдено. Такий підхід забезпечує зв'язок між вебінтерфейсом, серверною частиною та базою даних SQLite.

Клієнтська частина також містить функцію, яка відповідає за надсилання штрих-коду на серверну частину та обробку відповіді. Цей фрагмент коду показує зв'язок між інтерфейсом користувача та API пошуку товарів.

Лістинг 3.4 - Надсилання штрихкоду з клієнтської частини на сервер

```

const findProductByBarcode = async (code) => {
    if (!code) {
        setError("Введіть або відскануйте штрихкод товару.");
        setProduct(null);
        setAiAnalysis("");
        return;
    }
    try {
        setError("");
        setProduct(null);
        setAiAnalysis("");
        setBarcode(code);
    }
}

```

```

const response = await fetch(`/api/products/${code}`);
if (!response.ok) {
  const data = await response.json();
  setError(data.message || "Товар не знайдено.");
  return;
}
const data = await response.json();
setProduct(data);
} catch (err) {
  setError("Помилка з'єднання з сервером.");
}
};

```

У лістингу 3.4 показано функцію за клієнта, яка шукає товар по штрихкоду. Спочатку вона перевіряє, чи було передано код штрих-коду. Якщо поле порожнє, користувач пропонує ввести або відсканувати код. Якщо штрих-код отримано, фронтенд відправляє HTTP-запит на маршрут `/api/products/:barcode`. Якщо відповідь успішна, дані про товар зберігаються у стані товару та відображаються в інтерфейсі. Якщо товар не знайдено або недоступний сервер, користувач бачить відповідне повідомлення про помилку (див. лістинг 3.5).

Лістинг 3.5 – Приклад відповіді сервера

```

{
  "id": 1,
  "barcode": "4820000000012",
  "name": "Молоко домашнє 2.5%",
  "composition": "Молоко коров'яче нормалізоване, вітамін D.",
  "manufacturer": "ТОВ Молочний продукт",
  "category": "Молочні продукти"
}

```

А якщо товар не знайдено, то бекенд повертає повідомлення про помилку (лістинг 3.6):

Лістинг 3.6 – Повідомлення про помилку

```
{  
  "message": "Товар з таким штрихкодом не знайдено",  
  "barcode": "4820000000999"  
}
```

Окрім пошуку товарів, бекенд може додавати нові товари до бази даних. Це робиться за допомогою маршруту: “POST /api/products”.

Тіло запиту містить дані про товар: штрих-код, назву, склад, виробника та категорію. Сервер перевіряє обов'язкові поля, а потім додає запис до таблиці products.

Таким чином, взаємодія з СКБД реалізується через бекенд-рівень. Це дозволяє відокремити інтерфейс користувача від логіки бази даних, спрощуючи обслуговування системи та підвищуючи безпеку.

3.5 Використання технологій ШІ

У розробленій системі для аналізу складників продукту використовуються технології штучного інтелекту. Основна функція модуля ШІ полягає не в ідентифікації продукту за зображенням, а в поясненні вже отриманого текстового складників. Такий підхід є більш надійним для прототипу диплома, оскільки ідентифікація продукту здійснюється за штрих-кодом, а штучний інтелект використовується як допоміжний інструмент для аналізу інформації.

Після того, як продукт знайдено в базі даних, користувач бачить його склад. Якщо користувач хоче отримати додаткове пояснення, він натискає кнопку аналізу складників.

Лістинг 3.7 – Запуск AI-аналізу з клієнтської частини

```
const analyzeComposition = async () => {
  if (!product) {
    setError("Спочатку знайдіть товар.");
    return;
  }
  try {
    setIsAnalyzing(true);
    setError("");
    setAiAnalysis("");
    const response = await fetch("/api/analyze-composition", {
      method: "POST",
      headers: {
        "Content-Type": "application/json"
      },
      body: JSON.stringify({
        name: product.name,
        composition: product.composition
      })
    });
    const data = await response.json();
    if (!response.ok) {
      setError(data.message || "Не вдалося виконати AI-аналіз.");
      return;
    }
    setAiAnalysis(cleanAiText(data.analysis));
  } catch (err) {
    setError("Помилка з'єднання з AI-сервісом.");
  } finally {
    setIsAnalyzing(false);
  }
};
```

Фронтенд надсилає назву продукту та його склад до бекенду. Бекенд генерує текстовий запит до Gemini API. У запиті вказується, що необхідно

проаналізувати склад українською мовою, надати короткий висновок, визначити потенційні алергени, описати компоненти, на які варто звернути увагу, та сформулювати загальну рекомендацію.

У відповідь Gemini API повертає текстовий результат. Цей результат не є медичним висновком, а має інформаційний характер. Наприклад, модуль ШІ може вказати на те, що продукт містить молочні компоненти, сою або велику кількість цукру, тому людям з відповідними обмеженнями слід звернути на це увагу.

У той самий час необхідно враховувати обмеження аналізу ШІ. Він може помилятися чи формувати відповідь, ґрунтуючись на загальних знаннях, тому його висновки не слід сприймати як медичну рекомендацію. Саме тому доцільно використовувати в системі ретельне формулювання та вказувати, що аналіз має довідковий характер.

Таким чином, використання API Gemini дозволяє розширити функціональність системи та зробити її більш корисною для користувача. Модуль штучного інтелекту доповнює базову логіку пошуку товару за штрихкодом та надає інтелектуальне пояснення складників продукту.

Для виконання аналізу складників продукту в серверній частині реалізовано окремий API-маршрут. Він приймає назву продукту та його склад, формує текстовий запит до Gemini API та повертає результат аналізу до клієнтської частини. Відповідний фрагмент показано в лістингу 3.8.

Лістинг 3.8 – Аналіз складників товару за допомогою Gemini API

```
app.post("/api/analyze-composition", async (req, res) => {
  const { name, composition } = req.body;
  const prompt = `
    Проаналізуй склад товару українською мовою.
    Назва товару: ${name}
    Склад: ${composition}
    Вкажи потенційні алергени, важливі компоненти
    та дай коротку рекомендацію.
```

```

`;
const response = await ai.models.generateContent({
  model: "gemini-1.5-flash",
  contents: prompt,
});
res.json({
  analysis: response.text,
});
});

```

У цьому фрагменті бекенд отримує назву продукту та його склад від клієнтської сторони. На основі цих даних формується текстовий запит до Gemini API. У запиті зазначено, що аналіз має бути виконаний українською мовою, а результат має містити потенційні алергени, важливі компоненти та коротку рекомендацію. Після отримання відповіді від сервісу штучного інтелекту сервер повертає результат у форматі JSON, після чого він відображається у вебінтерфейсі.

Оскільки відповідь сервісу ШІ може містити символи форматування, у клієнтській частині реалізовано додаткову функцію для очищення тексту перед його відображенням.

Лістинг 3.9 – Очищення текстової відповіді ШІ-сервісу

```

const cleanAiText = (text) => {
  return text
    .replace(/\*\*/g, "")
    .replace(/\^s*\s?/gm, "• ")
    .replace(/\n{3,}/g, "\n\n")
    .trim();
};

```

У лістингу 3.9 показано функцію `cleanAiText`, яка використовується для підготовки відповіді сервісу штучного інтелекту до відображення у вебінтерфейсі. Функція видаляє непотрібні символи форматування, такі як подвійні

зірочки, перетворює позначені рядки у зручніший для користувача формат та видаляє зайві розриви рядків. Це робить результат аналізу більш читабельним для користувача.

4 ПРОЄКТУВАННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА

4.1 Опис інтерфейсу вебзастосунку

Головна сторінка вебзастосунку має просту структуру та складається з кількох основних блоків. У верхній частині сторінки знаходиться назва системи та короткий опис її призначення. Цей блок виконує інформаційну функцію та пояснює користувачеві, що система призначена для ідентифікації товарів за штрих-кодом та аналізу їхнього складників.

Основна частина сторінки поділена на дві функціональні області. Перша область призначена для введення або сканування штрих-коду. Друга область використовується для відображення результатів пошуку та аналізу за допомогою штучного інтелекту.

Блок введення даних містить кнопку запуску сканера, поле для ручного введення штрих-коду, кнопку пошуку та тестові кнопки з прикладами товарів. Кнопка запуску сканера дозволяє активувати камеру пристрою. Поле ручного введення використовується, якщо користувач хоче самостійно ввести штрих-код або якщо камера недоступна. Тестові кнопки потрібні для швидкої демонстрації роботи системи під час перевірки дипломного проєкту.

Блок результатів призначений для відображення інформації про знайдений товар. Якщо товар знайдено в базі даних, користувач бачить його назву, категорію, виробника, штрих-код та склад. Якщо товар не знайдено, система видає повідомлення про помилку. Такий підхід дозволяє користувачеві одразу зрозуміти результат виконаної дії.

Клієнтська частина вебзастосунку отримує доступ до серверної частини після введення або сканування штрих-коду. Це робиться за допомогою функції `fetch`, яка надсилає HTTP-запит до маршруту API пошуку товарів. Приклад такого запиту показано в лістингу 4.1.

Лістинг 4.1 – Запит клієнтської частини до backend

```
const response = await fetch(`/api/products/${code}`);
if (!response.ok) {
  const data = await response.json();
  setError(data.message || "Товар не знайдено.");
  return;
}
const data = await response.json();
setProduct(data);
```

У наведеному вище фрагменті коду змінна містить штрих-код, введений користувачем або отриманий після сканування. Клієнтська частина надсилає запит до бекенду за маршрутом `/api/products/:barcode`. Якщо сервер повертає помилку, повідомлення для користувача записується в стан помилки. Якщо товар знайдено, отримані дані записуються в стан товару та відображаються в блоці результатів вебінтерфейсу. Клієнтська частина веб-застосунку реалізована за допомогою бібліотеки `React`, яка призначена для створення компонентних користувацьких інтерфейсів.

Окремим елементом інтерфейсу є кнопка для запуску аналізу запасів за допомогою штучного інтелекту. Вона відображається після того, як товар знайдено. Після натискання цієї кнопки система надсилає інформацію про запаси товару на сервер, виконує запит до Gemini API та повертає результат аналізу. Отриманий текст відображається у спеціальному блоці, який візуально відокремлений від основної інформації про товар.

Для дизайну інтерфейсу було використано сучасний стиль з картками, заокругленими кутами, тінями та виділеними кнопками. Такий підхід робить

сторінку візуально зрозумілою та легкою для сприйняття. Основні елементи керування мають достатній розмір, щоб користувач міг легко взаємодіяти з ними як на комп'ютері, так і на мобільному пристрої.

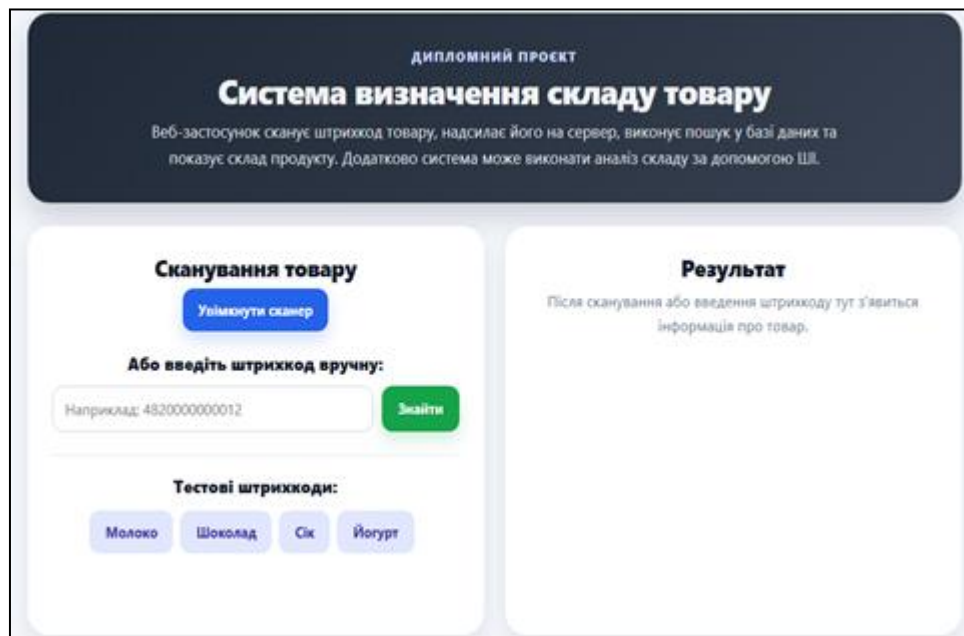


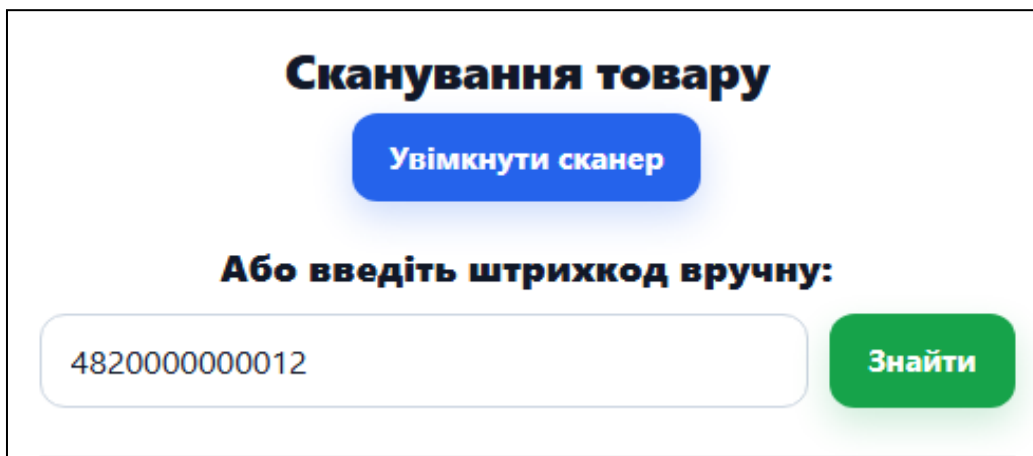
Рисунок 4.1 – Головна сторінка вебзастосунку

4.2 Основні екрани системи

Під час роботи користувач взаємодіє з кількома основними станами інтерфейсу. Кожен із цих станів відповідає певному етапу роботи системи.

Початковий екран відображається після відкриття вебзастосунку. На цьому екрані користувач бачить основні елементи керування, але інформація про продукт ще не відображається. У блоці результатів може відображатися запит про те, що потрібно ввести або відсканувати штрих-код.

Користувач вводить код продукту в текстове поле та натискає кнопку пошуку. Після цього система робить запит до бекенду. Цей стан зручний для тестування, оскільки дозволяє перевірити роботу системи без камери.



Сканування товару

Увімкнути сканер

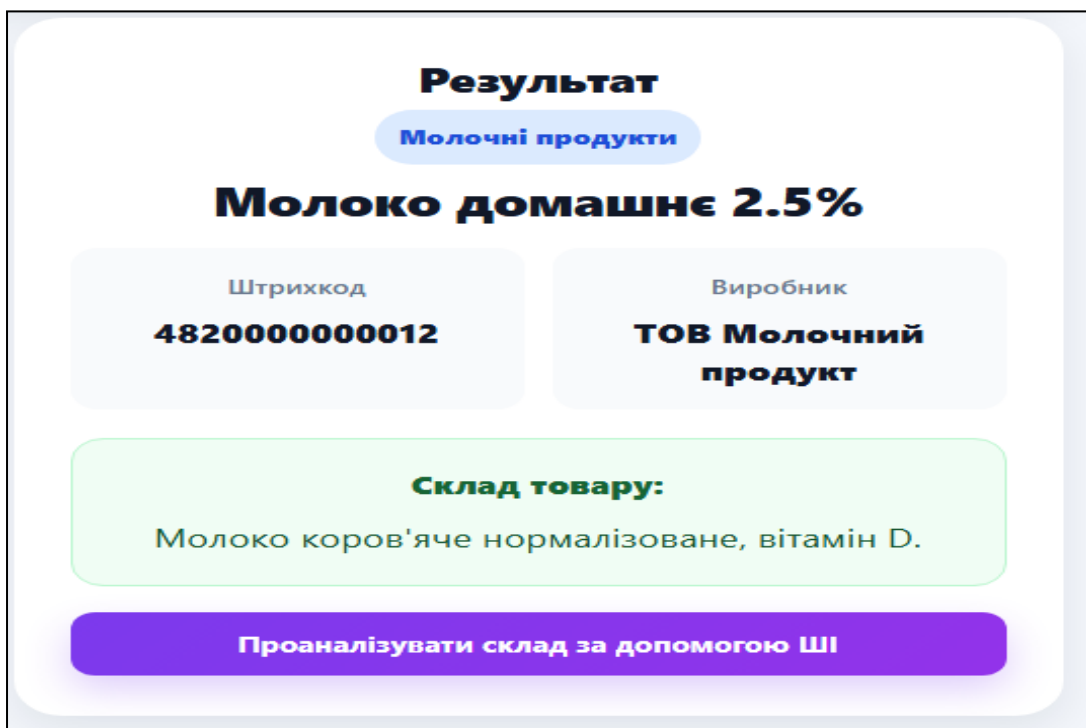
Або введіть штрихкод вручну:

4820000000012

Знайти

Рисунок 4.2 – Ручне введення штрихкоду

Якщо товар є у базі даних, система показує інформаційну картку з його даними. У цій картці відображаються назва товару, категорія, виробник, штрихкод та склад. Цей екран підтверджує, що frontend успішно передає запит на backend, а backend отримує інформацію з бази даних.



Результат

Молочні продукти

Молоко домашнє 2.5%

Штрихкод
4820000000012

Виробник
ТОВ Молочний продукт

Склад товару:
Молоко коров'яче нормалізоване, вітамін D.

Проаналізувати склад за допомогою ШІ

Рисунок 4.3 – Відображення інформації про знайдений товар

Після знаходження продукту користувач натискає кнопку аналізу складників. В очікуванні відповіді система може відобразити повідомлення про

обробку запиту. Після отримання результату на сторінці з'являється блок з аналізом складників. У цьому блоці відображається короткий висновок, потенційні алергени, компоненти, на які слід звернути увагу, та загальна рекомендація.

AI

Аналіз складу за допомогою ШІ

1. Короткий висновок про склад.

Склад Молока домашнього 2.5% є дуже простим і складається з основних компонентів, що очікуються від молока, з додаванням вітаміну. Нормалізація молока означає приведення його жирності до певного стандарту.

2. Потенційні алергени.

Основним потенційним алергеном у цьому продукті є коров'яче молоко. Це важливо для людей з алергією на молочний білок або непереносимістю лактози.

3. Компоненти, на які варто звернути увагу.

Варто звернути увагу на наявність вітаміну D, який часто додають до молока для збагачення. Цей вітамін є важливим для підтримки здоров'я кісток та імунної системи. Для людей з чутливістю до лактози або алергією на білок коров'ячого молока, цей продукт може бути невідповідним.

4. Загальна рекомендація.

Якщо у вас немає алергії на молочні продукти або непереносимості лактози, Молоко домашнє 2.5% може бути доброю частиною збалансованого харчування. Як і з будь-яким продуктом, рекомендується враховувати власні дієтичні потреби та можливі індивідуальні реакції.

Рисунок 4.4 – Результат аналізу ШІ складників товару

Якщо користувач вводить штрих-код, якого немає в базі даних, система відображає повідомлення про те, що продукт не знайдено. Цей екран важливий, оскільки він демонструє, як правильно обробляються помилкові ситуації.

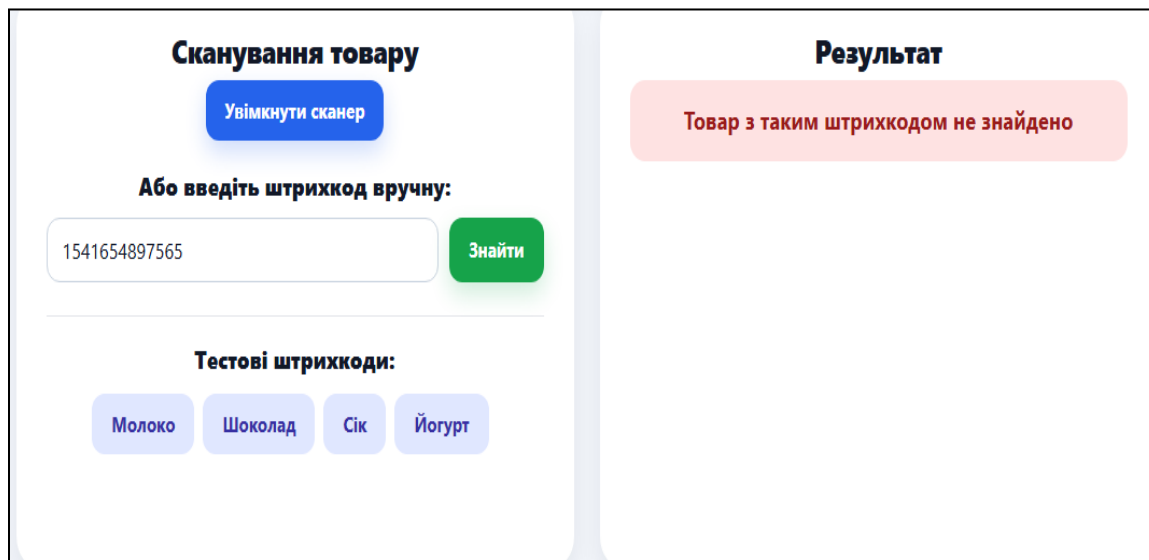


Рисунок 4.5 – Повідомлення про відсутність товару у БД

4.3 Результати роботи системи

В результаті розробки було створено прототип інформаційної системи для ідентифікації та відображення складників товарів. Система реалізована як клієнт-серверний вебзастосунок, що дозволяє користувачеві отримувати інформацію про товар за його штрих-кодом.

Розроблена система виконує такі основні функції:

- ручне введення штрих-коду товару;
- сканування штрих-коду за допомогою камери пристрою;
- передача штрих-коду на серверну частину;
- пошук товару в базі даних SQLite;
- відображення назви, виробника, категорії, штрих-коду та складників товару;

- аналіз складників товару за допомогою Gemini API;
- відображення результату ШІ-аналізу у вебінтерфейсі;
- обробка ситуацій, коли товар не знайдено або сервіс недоступний.

Практичним результатом роботи є створення робочого прототипу, який демонструє основні принципи функціонування сучасної інформаційної системи. У проєкті реалізовано фронтенд-частину, бекенд-частину, базу даних та інтеграцію із зовнішнім ШІ-сервісом.

Клієнтська частина забезпечує зручну взаємодію користувача з системою. Серверна частина обробляє запити, працює з базою даних та звертається до Gemini API. База даних використовується для зберігання інформації про продукти. Модуль штучного інтелекту дозволяє доповнити звичне відображення складників продукту поясненням у зрозумілій текстовій формі.

ВИСНОВКИ

Під час реалізації дипломного проєкту було розроблено прототип інтелектуальної інформаційної системи для ідентифікації та відображення складників товарів. Основною метою роботи було створення вебдодатку, який дозволяє користувачеві отримувати інформацію про продукт за його штрих-кодом, переглядати склад продукту та, за необхідності, проводити додатковий аналіз складників за допомогою технологій штучного інтелекту.

В результаті дипломного проєкту було реалізовано робочий прототип системи, яка виконує такі функції:

- дозволяє вводити штрих-код товару вручну;
- дозволяє сканувати штрих-код за допомогою камери пристрою;
- передає штрих-код на серверну частину;
- шукає товар у базі даних SQLite;
- відображає інформацію про знайдений товар;
- показує склад товару;
- виконує аналіз складників за допомогою Gemini API;
- відображає результат аналізу за допомогою ШІ у вебінтерфейсі;
- обробляє ситуації, коли товар не знайдено або послуга недоступна.

Практичне значення розробленої системи полягає в тому, що вона демонструє можливість поєднання вебтехнологій, бази даних, сканування штрих-кодів та штучного інтелекту в одному програмному продукті. Розроблений прототип може бути використаний як основа для подальшої розробки більш повної системи аналізу продуктів.

У майбутньому систему можна розширити шляхом додавання більшої кількості продуктів до бази даних, підключення зовнішніх джерел інформації про продукти, створення особистого облікового запису користувача, збереження історії сканування, додавання персональних обмежень щодо алергенів та впровадження мобільної версії застосунку.

Таким чином, мету дипломного проекту було досягнуто. Було розроблено інформаційну систему, яка забезпечує ідентифікацію продуктів за штрих-кодом, відображення їх складників та додатковий аналіз за допомогою технологій штучного інтелекту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. На заміну ДСТУ3008-95 ; чинний від 2017-07-01. Вид. офіц. Київ : УкрНДНЦ, 2016. 32 с.
2. ДСТУ 8302:2015. Інформація і документація. Бібліографічне посилання. Загальні положення та правила складання. Чинний від 2016-07-01. Вид. офіц. Київ : УкрНДНЦ, 2016. 30 с.
3. Нові бланки НУ «Запорізька політехніка». *Запорізька політехніка*. URL: <https://zp.edu.ua/novi-blanki-zntu/> (дата звернення: 18.04.2026).
4. Приклади оформлення посилань за ДСТУ 8302:2015 у списку використаних джерел. *Grafiati*. URL: <https://www.grafiati.com/uk/info/dstu-8302-2015/examples/> (дата звернення: 19.04.2026).
5. React Documentation. Офіційна документація бібліотеки React. URL: <https://react.dev/> (дата звернення: 26.04.2026).
6. Node.js Documentation. Офіційна документація середовища виконання Node.js. URL: <https://nodejs.org/en/docs> (дата звернення: 06.05.2026).
7. Express.js Documentation. Офіційна документація фреймворку Express.js. URL: <https://expressjs.com/> (дата звернення: 15.05.2026).
8. SQLite Documentation. Офіційна документація системи керування базами даних SQLite. URL: <https://www.sqlite.org/docs.html> (дата звернення: 15.05.2026).
9. html5-qrcode. Документація бібліотеки для сканування QR-кодів та штрихкодів у браузері. URL: <https://www.npmjs.com/package/html5-qrcode> (дата звернення: 15.05.2026).
10. Gemini API Documentation. Офіційна документація Google Gemini API. URL: <https://ai.google.dev/gemini-api/docs>
11. Open Food Facts API Documentation. Документація відкритої бази даних харчових продуктів. URL: <https://openfoodfacts.github.io/openfoodfacts-server/api/>

12. ISO/IEC/IEEE 29148:2018. Systems and software engineering — Life cycle processes — Requirements engineering.

13. Катренко А. В., Пасічник В. В. Прийняття рішень: теорія та практика : підруч. Львів : Новий світ – 2000, 2023. 447 с.

14. Object Management Group® Standards Development Organization. URL: <https://www.omg.org/index.htm/> (дата звернення: 25.04.2026).

15. better-sqlite3. Документація npm-пакета для роботи з SQLite у Node.js. URL: <https://www.npmjs.com/package/better-sqlite3>

ДОДАТОК А

ЛІСТИНГИ ПРОГРАМ

Лістинг А.1 – Лістинг серверної частини server.js

```

require("dotenv").config();
const express = require("express");
const cors = require("cors");
const Database = require("better-sqlite3");
const { GoogleGenAI } = require("@google/genai");
const app = express();
const PORT = 5000;
const ai = new GoogleGenAI({
  apiKey: process.env.GEMINI_API_KEY,
});
app.use(cors());
app.use(express.json());
const db = new Database("products.db");
db.exec(`
  CREATE TABLE IF NOT EXISTS products (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    barcode TEXT UNIQUE NOT NULL,
    name TEXT NOT NULL,
    composition TEXT NOT NULL,
    manufacturer TEXT,
    category TEXT
  );
`);
const insertProduct = db.prepare(`
  INSERT OR IGNORE INTO products
    (barcode, name, composition, manufacturer, category)
  VALUES (?, ?, ?, ?, ?)
`);
insertProduct.run(
  "4820000000012",

```

```

        "Молоко домашнє 2.5%",
        "Молоко коров'яче нормалізоване, вітамін D.",
        "ТОВ Молочний продукт",
        "Молочні продукти"
    );
    insertProduct.run(
        "48200000000029",
        "Шоколад молочний",
        "Цукор, какао-масло, какао терте, сухе молоко,
        емульгатор соєвий лецитин, ароматизатор ванілін.",
        "ТОВ Солодкий світ",
        "Кондитерські вироби"
    );
    insertProduct.run(
        "48200000000036",
        "Сік яблучний",
        "Сік яблучний відновлений, вода питна, концентрований
        яблучний сік, регулятор кислотності лимонна кислота.",
        "ТОВ Фруктовий сад",
        "Напої"
    );
    insertProduct.run(
        "48200000000043",
        "Йогурт полуничний",
        "Молоко нормалізоване, закваска йогуртна, полуничний
        наповнювач, цукор, стабілізатор пектин.",
        "ТОВ Молочний продукт",
        "Молочні продукти"
    );
    app.get("/", (req, res) => {
        res.json({
            message: "API системи визначення складників товару
            працює",
        });
    });
}

```

```

app.get("/api/products", (req, res) => {
  const products = db.prepare("SELECT * FROM products
ORDER BY id DESC").all();
  res.json(products);
});

app.get("/api/products/:barcode", (req, res) => {
  const { barcode } = req.params;
  const product = db
    .prepare("SELECT * FROM products WHERE barcode = ?")
    .get(barcode);
  if (!product) {
    return res.status(404).json({
      message: "Товар з таким штрихкодом не знайдено",
      barcode,
    });
  }
  res.json(product);
});

app.post("/api/products", (req, res) => {
  const { barcode, name, composition, manufacturer,
category } = req.body;
  if (!barcode || !name || !composition) {
    return res.status(400).json({
      message: "Поля barcode, name та composition є
обов'язковими",
    });
  }
  try {
    const result = db
      .prepare(
        `
        INSERT INTO products
        (barcode, name, composition, manufacturer,
category)
        VALUES (?, ?, ?, ?, ?)
      `
      .execute()
  } catch (error) {
    console.error(error);
    return res.status(500).json({
      message: "Внутрішня помилка сервера"
    });
  }
  res.status(201).json({
    message: "Товар успішно додано"
  });
});

```

```

        )
        .run(barcode, name, composition, manufacturer || "",
category || "");
    res.status(201).json({
        message: "Товар успішно додано",
        id: result.lastInsertRowid,
    });
} catch (error) {
    res.status(500).json({
        message: "Помилка при додаванні товару",
        error: error.message,
    });
}
});
app.post("/api/analyze-composition", async (req, res) => {
    const { name, composition } = req.body;
    if (!composition) {
        return res.status(400).json({
            message: "Склад товару є обов'язковим для аналізу.",
        });
    }
    if (!process.env.GEMINI_API_KEY) {
        return res.status(500).json({
            message: "Gemini API key не налаштований у файлі
.env.",
        });
    }
    try {
        const prompt = `
Ти — помічник для аналізу складників харчових продуктів.

Проаналізуй склад товару українською мовою.

Назва товару: ${name || "невідомо"}

Склад товару:

```

`${composition}`

Потрібно дати коротку, зрозумілу відповідь для звичайного користувача.

ВАЖЛИВО:

- Не використовуй Markdown.
- Не використовуй символи **.
- Не використовуй жирний текст.
- Не використовуй таблиці.
- Відповідь має бути чистим текстом.

Структура відповіді:

1. Короткий висновок про склад.
2. Потенційні алергени.
3. Компоненти, на які варто звернути увагу.
4. Загальна рекомендація.

Не став медичний діагноз. Не пиши, що продукт точно шкідливий або точно корисний. Пиши обережно і нейтрально.

`;

```
const response = await ai.models.generateContent({
  model: "gemini-2.5-flash",
  contents: prompt,
});
res.json({
  analysis: response.text,
});
} catch (error) {
  console.error("Gemini error:", error);
  res.status(500).json({
    message: "Помилка під час аналізу складників за
допомогою ШІ.",
    error: error.message,
  });
}
});
app.listen(PORT, () => {
```

```

        console.log(`Backend запущено на
http://localhost:${PORT}`);
    });

```

Лістинг А.2 – Лістинг клієнтської частини App.jsx

```

import { useEffect, useRef, useState } from "react";
import { Html5QrcodeScanner } from "html5-qrcode";
import "./App.css";
function App() {
    const [barcode, setBarcode] = useState("");
    const [manualBarcode, setManualBarcode] = useState("");
    const [product, setProduct] = useState(null);
    const [error, setError] = useState("");
    const [isScannerVisible, setIsScannerVisible] =
useState(false);
    const [aiAnalysis, setAiAnalysis] = useState("");
    const [isAnalyzing, setIsAnalyzing] = useState(false);
    const scannerRef = useRef(null);
    const cleanAiText = (text) => {
        return text
            .replace(/\s*/g, "")
            .replace(/^\s*\s?/gm, "• ")
            .replace(/\n{3,}/g, "\n\n")
            .trim();
    };
    const findProductByBarcode = async (code) => {
        if (!code) {
            setError("Введіть або відскануйте штрихкод
товару.");
            setProduct(null);
            setAiAnalysis("");
            return;
        }
        try {
            setError("");

```

```

    setProduct(null);
    setAiAnalysis("");
    setBarcode(code);
    const response = await fetch(`/api/products/${code}`);
    if (!response.ok) {
        const data = await response.json();
        setError(data.message || "Товар не знайдено.");
        return;
    }
    const data = await response.json();
    setProduct(data);
} catch (err) {
    setError("Помилка з'єднання з сервером.");
}
};

const analyzeComposition = async () => {
    if (!product) {
        setError("Спочатку знайдіть товар.");
        return;
    }
    try {
        setIsAnalyzing(true);
        setError("");
        setAiAnalysis("");
        const response = await fetch("/api/analyze-
composition", {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify({
                name: product.name,
                composition: product.composition
            })
        });
        const data = await response.json();

```

```

        if (!response.ok) {
            setError(data.message || "Не вдалося виконати AI-аналіз.");
            return;
        }
        setAiAnalysis(cleanAiText(data.analysis));
    } catch (err) {
        setError("Помилка з'єднання з AI-сервісом.");
    } finally {
        setIsAnalyzing(false);
    }
};

const handleManualSearch = (event) => {
    event.preventDefault();
    findProductByBarcode(manualBarcode.trim());
};

const startScanner = () => {
    setIsScannerVisible(true);
    setError("");
    setProduct(null);
    setAiAnalysis("");
};

const stopScanner = async () => {
    if (scannerRef.current) {
        try {
            await scannerRef.current.clear();
        } catch (error) {
            console.log("Scanner clear error:", error);
        }
        scannerRef.current = null;
    }
    setIsScannerVisible(false);
};

useEffect(() => {
    if (!isScannerVisible) return;
    const scanner = new Html5QrcodeScanner(

```

```

    "reader",
    {
      fps: 10,
      qrbox: {
        width: 250,
        height: 250
      }
    },
    false
  );
  scannerRef.current = scanner;
  scanner.render(
    async (decodedText) => {
      await stopScanner();
      setManualBarcode(decodedText);
      findProductByBarcode(decodedText);
    },
    (scanError) => {
      console.log(scanError);
    }
  );
  return () => {
    if (scannerRef.current) {
      scannerRef.current.clear().catch(() => {});
      scannerRef.current = null;
    }
  };
}, [isScannerVisible]);
return (
  <div className="page">
    <header className="header">
      <div>
        <p className="label">Дипломний проект</p>
        <h1>Система визначення складників товару</h1>
        <p className="description">

```

Вебзастосунок сканує штрихкод товару, надсилає його на сервер, виконує пошук у базі даних та показує склад продукту. Додатково

система може виконати аналіз складників за допомогою ШІ.

```

    </p>
  </div>
</header>
<main className="content">
  <section className="card">
    <h2>Сканування товару</h2>

    <div className="buttons">
      {!isScannerVisible ? (
        <button className="primary-button"
onClick={startScanner}>
          Увімкнути сканер
        </button>
      ) : (
        <button className="secondary-button"
onClick={stopScanner}>
          Вимкнути сканер
        </button>
      )}
    </div>

    {isScannerVisible && (
      <div className="scanner-box">
        <div id="reader"></div>
      </div>
    )}

    <form className="manual-form"
onSubmit={handleManualSearch}>
      <label htmlFor="barcode">Або введіть штрихкод
вручну:</label>

      <div className="input-row">
        <input

```

```

        id="barcode"
        type="text"
        value={manualBarcode}
        onChange={ (event) =>
setManualBarcode(event.target.value) }
        placeholder="Наприклад: 4820000000012"
      />
      <button type="submit">Знайти</button>
    </div>
  </form>
  <div className="test-codes">
    <p>Тестові штрихкоди:</p>
    <button onClick={ () =>
findProductByBarcode("4820000000012") }>
      Молоко
    </button>
    <button onClick={ () =>
findProductByBarcode("4820000000029") }>
      Шоколад
    </button>
    <button onClick={ () =>
findProductByBarcode("4820000000036") }>
      Сік
    </button>
    <button onClick={ () =>
findProductByBarcode("4820000000043") }>
      Йогурт
    </button>
  </div>
</section>
<section className="card result-card">
  <h2>Результат</h2>
  {!product && !error && (
    <p className="empty-text">
      Після сканування або введення штрихкоду тут
      з'явиться інформація

```

```

        про товар.
    </p>
  ) }
  {error && <div className="error">{error}</div>}
  {product && (
    <div className="product">
      <p className="product-
category">{product.category}</p>
      <h3>{product.name}</h3>
      <div className="info-grid">
        <div>
          <span>Штрихкод</span>
          <strong>{product.barcode}</strong>
        </div>
        <div>
          <span>Виробник</span>
          <strong>{product.manufacturer}</strong>
        </div>
      </div>
      <div className="composition">
        <h4>Склад товару:</h4>
        <p>{product.composition}</p>
      </div>
      <button
        className="ai-button"
        onClick={analyzeComposition}
        disabled={isAnalyzing}
      >
        {isAnalyzing
          ? "ШІ аналізує склад..."
          : "Проаналізувати склад за допомогою
ШІ"}
      </button>
    {aiAnalysis && (
      <div className="ai-analysis">
        <div className="ai-title-row">

```

```

        <div className="ai-icon">AI</div>
        <h4>Аналіз складників за допомогою
III</h4>

        </div>
        <div className="ai-text">
            {aiAnalysis.split("\n").map((line,
index) => {

                if (!line.trim()) {
                    return <br key={index} />;
                }
                return <p key={index}>{line}</p>;
            })}
        </div>
    </div>
    )}
</div>
    )}
</section>
</main>
</div>
);
}
export default App;

```

Лістинг A.3 – Лістинг конфігурації vite.config.js

```

import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";
export default defineConfig({
  plugins: [react()],
  server: {
    proxy: {
      "/api": "http://localhost:5000",
    },
  },
});

```

ДОДАТОК Б

СЛАЙДИ ПРЕЗЕНТАЦІЇ

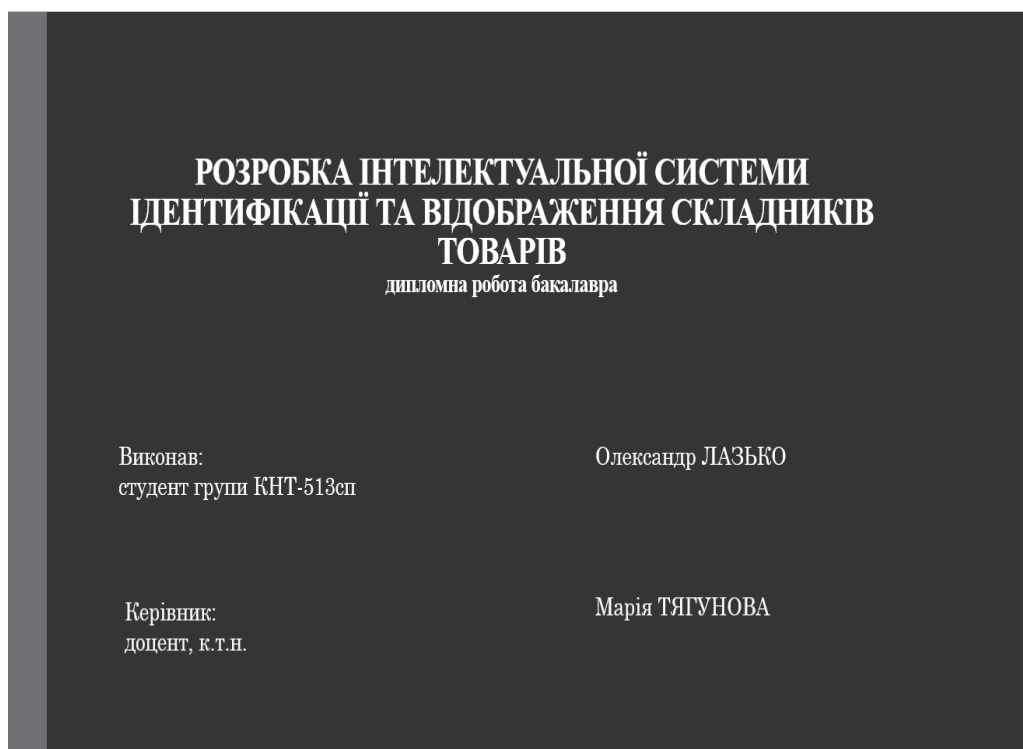


Рисунок Б.1 – Слайд 1

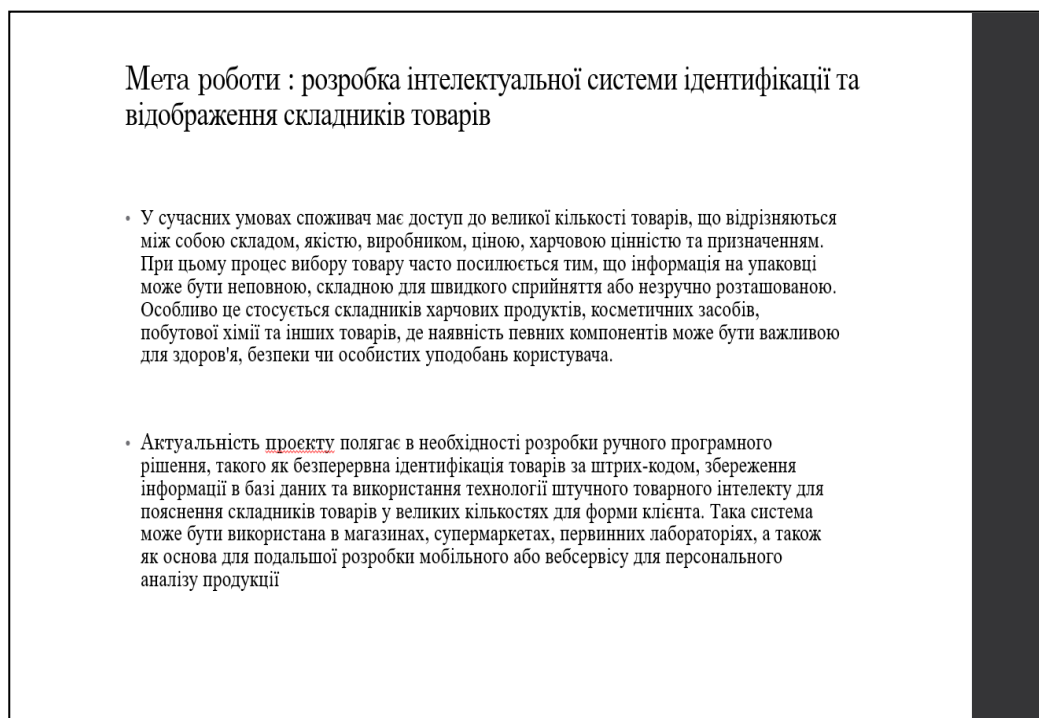


Рисунок Б.2 – Слайд 2



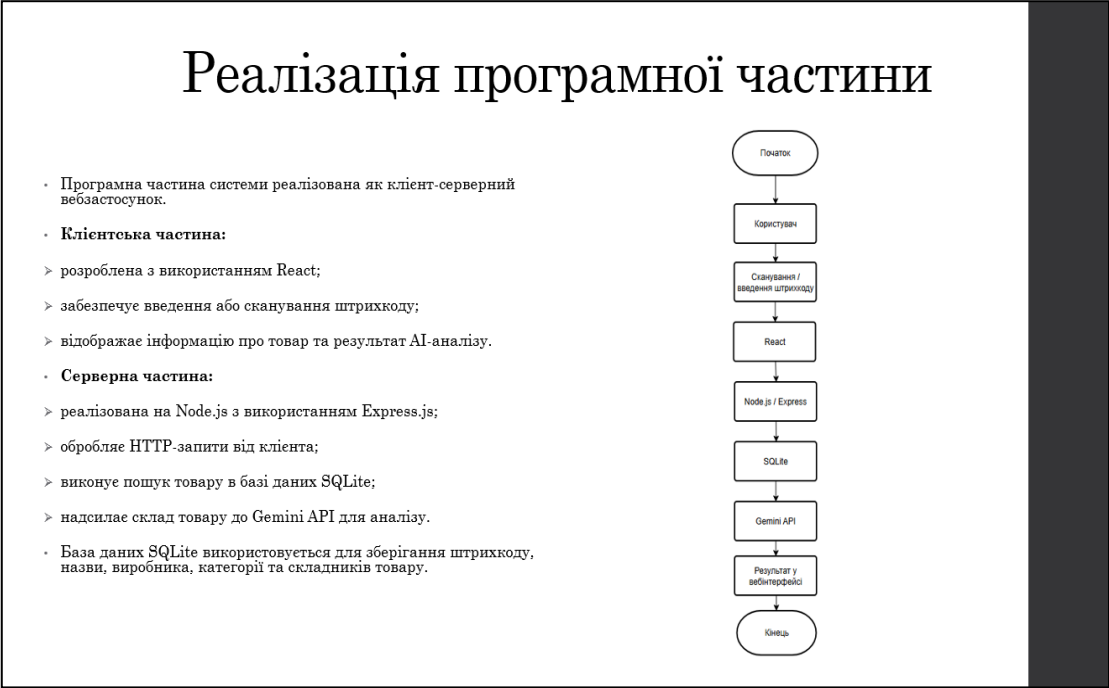


Рисунок Б.5 – Слайд 5

Результати роботи системи

Головна сторінка вебзастосунку

Система визначення складу товару
Веб-застосунок пошуку складу товару, надання його на основі введення штрихкоду та показу складу продукту. Додатково система може виконати аналіз складу за допомогою ШІ.

Сканування товару
Введіть штрихкод або введіть штрихкод вручну:
Штрихкод: 4820000000012

Результат
Після сканування ШІ проаналізує склад продукту та надішле інформацію про товар.

Результат аналізу ШІ складників товару

Аналіз складу за допомогою ШІ

1. Короткий висновок про склад.
Склад Молока домашнього 2.5% є дуже простим і складається з основних компонентів, що очікуються від молока, з додаванням вітаміну. Нормалізація молока означає приведення його жирності до певного стандарту.

2. Потенційні алергени.
Основним потенційним алергеном у цьому продукті є коров'яче молоко. Це важливо для людей з алергією на молочний білок або непереносимістю лактози.

3. Компоненти, на які варто звернути увагу.
Варто звернути увагу на наявність вітаміну D, який часто додають до молока для збагачення. Цей вітамін є важливим для підтримки здоров'я кісток та імунної системи. Для людей з чутливістю до лактози або алергією на білок коров'ячого молока, цей продукт може бути невідповідним.

4. Загальна рекомендація.
Якщо у вас немає алергії на молочні продукти або непереносимості лактози, Молоко домашнє 2.5% може бути доброю частиною збалансованого харчування. Як і з будь-яким продуктом, рекомендується враховувати власні дієтичні потреби та можливі індивідуальні реакції.

Відображення інформації про знайдений товар

Результат
Молочні продукти

Молоко домашнє 2.5%

Штрихкод: 4820000000012 | Виробник: ТОВ Молочний продукт

Склад товару:
Молоко коров'яче нормалізоване, вітамін D.

Проманіпулювати склад за допомогою ШІ

Рисунок Б.6 – Слайд 6

Висновки

- В результаті дипломного проекту було реалізовано робочий прототип системи, яка виконує такі функції:
 - дозволяє вводити штрих-код товару вручну;
 - дозволяє сканувати штрих-код за допомогою камери пристрою;
 - передає штрих-код на серверну частину;
 - шукає товар у базі даних SQLite;
 - відображає інформацію про знайдений товар;
 - показує склад товару;
 - виконує аналіз складників за допомогою Gemini API;
 - відображає результат аналізу за допомогою ШІ у вебінтерфейсі;
 - обробляє ситуації, коли товар не знайдено або послуга недоступна.
- Таким чином, мету дипломного проекту було досягнуто. Було розроблено інформаційну систему, яка забезпечує ідентифікацію продуктів за штрих-кодом, відображення їх складників та додатковий аналіз за допомогою технологій штучного інтелекту.

Рисунок Б.7 – Слайд 7