Міністерство освіти і науки України

Національний технічний університет
«Дніпровська політехніка»

Навчально-науковий
інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

здобувача Бєженцев Данило Олександрович
(ПІБ)

академічної групи 123-21-2
(шифр)

спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)

за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)

на тему "Веб-орієнтована комп'ютерна система з інтеграцією API для сканування штрих-кодів різних форматів"
(назва за наказом ректора)

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
кваліфікаційної роботи	доц. Каштан В.Ю.			
загального розділу	доц. Каштан В.Ю.			
спеціального розділу	доц. Каштан В.Ю.			
Рецензент				
Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2025

ЗАТВЕРДЖЕНО:
завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)
_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)

"25" січня 2025 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавр

здобувача Бєженцев Д.О. **академічної групи** 123-21-2
(прізвище та ініціали) (шифр)

спеціальності 123 Комп'ютерна інженерія

за освітньо-професійною програмою Комп'ютерна інженерія
(офіційна назва)

на тему " Веб-орієнтована комп'ютерна система з інтеграцією API для сканування штрих-кодів різних форматів"

затверджену наказом ректора НТУ «Дніпровська політехніка» від 05.05.2025 № 336-с

Розділ	Зміст	Термін виконання
Загальний розділ	На основі матеріалів виробничих практик, інших науково-технічних джерел показати актуальність завдання, сформулювати мету та задачі виконання кваліфікаційної роботи	10.02.2025
Спеціальний розділ	Сформулювати найменування й призначення веб-орієнтованної системи, висунути технічні вимоги до нього	15.03.2025
	Розробити архітектуру веб-орієнтованної системи, обґрунтування технічних характеристик.	20.04.2025
	Реалізувати програмний модуль, використовуючи відповідні інструменти програмування та враховуючи принципи розробки надійних та ефективних програмних систем, що є частиною комп'ютерної інженерії	07.05.2025
	Протестувати розроблену веб-орієнтовану систему на предмет її функціональності, продуктивності обміну даними в реальному часі та стійкості до можливих збоїв	31.05.2025

Завдання видано _____ доц. Каштан В.Ю.
(підпис керівника) (прізвище, ініціали)

Дата видачі 25.01.2025

Дата подання до екзаменаційної комісії _____

Прийнято до виконання _____ Бєженцев Д.О.

РЕФЕРАТ

Пояснювальна записка: 75 с., 22 рис., 1 табл., 1 додаток, 20 джерел.

Об'єктом дослідження є веб-орієнтовані комп'ютерні системи, зокрема програмні рішення для інтеграції API для сканування штрих-кодів та QR-кодів з метою автоматизації обробки товарної інформації в реальному часі.

Предметом дослідження є програмне забезпечення для сканування та обробки штрих-кодів різних форматів, інтегроване з базою даних для зберігання та управління інформацією про товари. Основна увага зосереджена на розробці інтерфейсу для зручного введення та відображення даних, а також на розробці системи, що підтримує різноманітні методи сканування, включаючи завантаження зображень, використання камери та вручну введення штрих-кодів.

Метою роботи є розробка веб-орієнтованої комп'ютерної системи для сканування штрих-кодів різних форматів, з інтеграцією API для обробки ідентифікації товарів, з можливістю подальшого збереження результатів у базу даних. Система повинна забезпечувати швидке та зручне введення інформації, ефективне використання камери для сканування в реальному часі та інтеграцію з базою даних для відображення актуальної інформації про товари.

Завдання:

1. Розробити інтерфейс користувача для зручного введення та відображення результатів сканування штрих-кодів та QR-кодів.

2. Створити базу даних для зберігання інформації про товари, включаючи штрих-коди, назви товарів, ціни та кількість.

3. Реалізувати можливість сканування штрих-кодів з використанням камери, а також зображень із файлів.

4. Розробити механізм для перевірки наявності товару в базі даних та відображення результатів користувачеві.

5. Забезпечити можливість додавання товарів до кошика з оновленням даних про кількість.

Ключові слова: штрих-коди, база даних, комп’ютерна система, API.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК, ОДИНИЦЬ ТА ТЕРМІНІВ

AIDC	–	автоматична ідентифікація даних;
RFID	–	радіочастотна ідентифікація;
QR	–	Quick Response Code;
CCD	–	Charge-Coupled Device;
UML	–	Unified Modeling Language;
REST	–	Representational State Transfer;
MVC	–	Model-View-Controller;
БД	–	база даних;
API	–	Application Programming Interface.

ВСТУП

У світі цифровізації та автоматизації все більшу роль відіграють технології автоматичної ідентифікації даних (AIDC), зокрема штрих-коди, радіочастотна ідентифікація (RFID), смарт-картки та інші засоби збору даних. Серед них саме штрих-коди залишаються найпоширенішим способом маркування товарів уже понад п’ять десятиліть. Їх широке використання пояснюється простотою друку, швидкістю сканування та дешевизною впровадження. Чорно-білі смуги зчитуються сканерами і миттєво дозволяють отримати інформацію про продукт – його назву, ціну, залишок, дату виготовлення тощо.

Застосування штрих-кодів охоплює практично всі сфери економіки – від роздрібної торгівлі до логістики, від сільського господарства до фармацевтики. Зокрема, в системах садівництва та квіткового виробництва штрих-коди використовуються для ідентифікації лотків із живцями, фіксації даних про посадковий матеріал, обліку залишків, автоматизації перерахунку та повторного замовлення. Проте традиційні лазерні сканери, які потребують прямої видимості та фізичної близькості до коду, мають значні обмеження. Вони не справляються з пошкодженими, вицвілими або брудними штрих-кодами, що часто трапляється в реальних умовах виробництва, особливо в тепличних господарствах чи на складах.

В умовах, коли точність і швидкість сканування набувають критичного значення для ефективності процесів, виникає потреба в нових підходах – гнучких, доступних, незалежних від дорогого обладнання. Сучасні технології обробки зображень, зокрема ті, що базуються на бібліотеках OpenCV, PIL, а також доступність веб-інтерфейсів і API, відкривають можливість створення адаптивних комп’ютерних систем для розпізнавання штрих-кодів з використанням звичайних камер або завантажених зображень. Такі системи можна впроваджувати в роздрібну торгівлю, логістику, виробництво та інші галузі, де автоматизація обліку є ключовою.

Саме тому у цій кваліфікаційній роботі запропоновано створення веб-орієнтованої комп’ютерної системи з інтеграцією API для сканування штрих-кодів різних форматів. Розроблена система дозволяє користувачам розпізнавати штрих-

коди за допомогою веб-камери, завантажених зображень або ручного введення, перевіряти інформацію про товари у базі даних, додавати їх до умовного кошика, а також копіювати інформацію в буфер обміну для подальшої обробки. У системі реалізовано простий та інтуїтивно зрозумілий інтерфейс з використанням Python, бібліотек tkinter, OpenCV, PIL і sqlite3.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Проаналізувати сучасні підходи до сканування та розпізнавання штрих-кодів та QR-кодів.

2. Дослідити доступні API та програмні бібліотеки для роботи з графічним зображенням та камерою.

3. Розробити структуру бази даних для зберігання інформації про товари (назва, ціна, залишок, штрих-код).

4. Реалізувати функціонал розпізнавання штрих-кодів із зображення, веб-камери та при ручному введенні.

5. Здійснити інтеграцію з базою даних для перевірки коду та виводу інформації про знайдений товар.

6. Розробити графічний інтерфейс користувача, що дозволяє взаємодіяти із системою легко та інтуїтивно.

7. Здійснити тестування системи на коректність роботи, продуктивність та зручність використання

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Опис об’єкта розробки та умов застосування веб-орієнтованих систем

Штрих-коди є основним засобом автоматизації процесів обліку товарів, що активно використовуються в багатьох виробничих та логістичних процесах. Їх застосування дозволяє значно знизити людський фактор, покращити точність і швидкість обробки даних, зменшити час на інвентаризацію, а також оптимізувати облікові та управлінські процеси.

Зокрема, у сільському господарстві, де ефективність процесів безпосередньо залежить від точності і швидкості обробки інформації, системи автоматичної ідентифікації, які використовують штрих-коди, допомагають вирішити проблеми відстеження рослинного матеріалу на всіх етапах його обробки — від посадки до реалізації готової продукції. Вони дозволяють вести облік посадкових лотків, контейнерів з розсадою, а також матеріалів для вирощування квітів, що дає змогу підтримувати високий рівень управління запасами та знижувати витрати на організацію процесів.

Водночас проблема, яку має вирішити ця система, полягає в забезпеченні високої точності та швидкості зчитування штрих-кодів за допомогою звичайних камер, що дає змогу зменшити витрати на спеціалізоване обладнання (лазерні сканери) і зробити автоматизацію більш доступною для малих і середніх підприємств. Визначення штрих-кодів на зображеннях з камер на складах чи виробничих лініях може бути складним через різноманітні умови освітлення, зміщення об’єкта, різні кути зйомки, а також різну якість самих етикеток.

Оскільки штрих-код є частиною етикетки і може використовуватися як об'єкт виявлення, штрих-код запроваджено в систему виявлення. Існуючі підходи до виявлення штрих-кодів в основному працюють для великих за розміром штрих-кодів, які мають добре видимі і чіткі мітки. Однак у реальних умовах виробництва часто виникає потреба в роботі з маленькими етикетками, що мають складний фон або низьку контрастність, що ускладнює їх виявлення [1].

Завданням цієї роботи є розробка та впровадження комбінованого методу, який дозволить виявляти мітки малого розміру під великим складним фоном. Це включає в себе вирішення проблеми зчеплення зображення при різних умовах освітлення, зміщеннях, перекосах та різних кутах зйомки, а також зчитування міток за відстані від 1 до 4 метра. Проблеми, пов’язані з детекцією міток на зображеннях, будуть вирішуватися через застосування інваріантних до афінних перетворень методів, таких як обертання, викривлення та масштабування. Крім того, система повинна бути здатною працювати з різними типами етикеток, які можуть бути як одновимірними (1D), так і двовимірними (2D), а також вміти розпізнавати текстову інформацію на етикетках.

Задача полягає в поєднанні різних алгоритмів обробки зображень, щоб забезпечити одночасне розпізнавання штрих-кодів і текстів на етикетках. Це дозволить зберегти високу точність та швидкість процесу, навіть за умов складного фону або пошкоджених етикеток. Такий підхід дозволить значно покращити ефективність автоматизації процесів на складах і в логістичних системах, зменшуючи потребу в спеціалізованому обладнанні і забезпечуючи високу надійність і точність на всіх етапах обробки товарів.

1.2 Структура та технології штрих-кодової ідентифікації

Штрих-кодова ідентифікація є ключовою технологією автоматизованого збору даних, що використовується в широкому спектрі галузей, від роздрібної торгівлі до логістики та виробництва. Штрих-коди, по суті, є машиночитаним представленням даних, що зазвичай описують певні характеристики об’єкта, який їх містить. Сьогодні штрих-коди присутні практично всюди, від пакування харчових продуктів у супермаркетах до ідентифікаційних браслетів пацієнтів у лікарнях, існуючи в різноманітних формах та версіях.

Перш за все, штрих-коди можна розділити на дві основні категорії: одномірні (1D) та двовимірні (2D). Одномірний штрих-код складається з вертикальних ліній, розміщених у межах прямокутної області. Інформація в 1D штрих-коді кодується

шляхом варіювання проміжків між цими вертикальними лініями та їхньої ширини (рис.1.1) [2].

Рисунок 1.1 – Структура одновимірного штрих-коду: а) показує зміну відстані між вертикальними лініями; б) показує, як може змінюватися ширина вертикальних ліній

З іншого боку, двовимірні штрих-коди, на відміну від використання ліній, зберігають інформацію у вигляді сітки зі спеціальних квадратних елементів, які називаються модулями, розташованих як у вертикальному, так і в горизонтальному напрямках (рис. 1.2) [3].

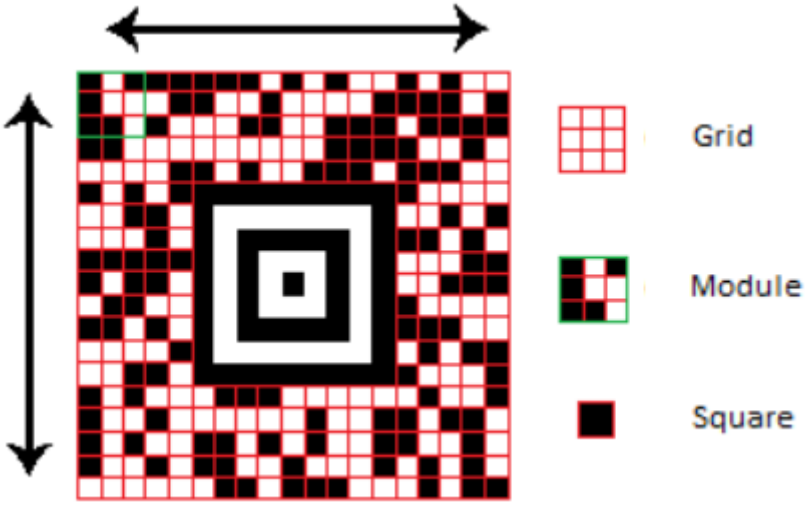

Рисунок 1.2 – Структура двовимірного штрих-коду

Двовимірні штрих-коди набувають все більшого поширення завдяки своїй

значно більшій ємності порівняно з 1D штрих-кодами. Це дозволяє зберігати різноманітні типи інформації, такі як текст, візитні картки (Vcard), уніфіковані локатори ресурсів (URL-адреси) та багато іншого. Серед двовимірних штрих-кодів особливе місце займає код швидкого реагування (QR), розроблений в Японії. QR-код відрізняється високою ємністю зберігання даних, зокрема завдяки своїй здатності кодувати символи кандзі. Крім QR-кодів, існують також інші поширені двовимірні формати, такі як Data Matrix, PDF417 та Aztec Codes.

QR-коди (Quick Response Code) – це тип двовимірного штрих-коду, який був вперше представлений компанією Denso Wave в 1994 році. Спочатку QR-коди використовувалися в автомобільній промисловості для відстеження транспортних засобів під час виробництва. Однак їх використання швидко стало набагато ширшим, і сьогодні QR-коди знаходять застосування в численних сферах: від маркетингу до медичних записів і електронних платіжних систем. Це стало можливим завдяки низці переваг, які QR-коди мають перед традиційними одновимірними штрих-кодами, включаючи високу швидкість зчитування та велику ємність для зберігання інформації [4].

QR-коди належать до 2D штрих-кодів, що означає, що вони здатні зберігати дані як по горизонталі, так і по вертикалі. Ця особливість дозволяє їм містити набагато більше інформації в порівнянні з одновимірними штрих-кодами, які зберігають лише лінійні дані. Зараз QR-коди широко використовуються для зберігання різноманітної інформації, такої як текст, посилання на веб-сторінки (URI), контакти в форматі vCard, а також для багатьох інших цілей.

QR-коди представлені у вигляді квадратної сітки, всередині якої розташовані маленькі чорні та білі модулі (точки), що формують візуальний патерн, який містить закодовану інформацію. Ця структура є строго визначеною і розподілена по зонах, кожна з яких виконує певну функцію в забезпеченні точного зчитування, декодування та корекції помилок.

Для прикладу розглянемо структуру QR-коду версії 10 (57x57 модулів), що представлено на рис.1.3.

Рисунок 1.3 – Структура QR-коду

Тиха зона (Quiet Zone) – це біла область, що оточує QR-код по периметру. Вона забезпечує візуальне відокремлення коду від оточення, що дозволяє сканеру ідентифікувати початок і кінець коду. Відсутність цієї зони може ускладнити або унеможливити зчитування.

Шаблони визначення положення (Finder Patterns) – три однакові квадратні шаблони, розташовані у верхньому лівому, верхньому правому та нижньому лівому кутах коду. Вони мають структуру 7x7 модулів і включають центральний чорний квадрат 3x3, обрамлений білим і чорним модулями. Їх призначення — дозволити сканеру швидко знайти QR-код у зображенні, а також визначити його орієнтацію (поворот).

Шаблони вирівнювання (Alignment Patterns) – використовуються в QR-кодах версій 2 і вище для компенсації можливих викривлень при скануванні, зокрема нахилу або викривлення перспективи. Вони допомагають точніше інтерпретувати позицію інших модулів у сітці.

Шаблон синхронізації (Timing Pattern) – складається з чергування чорних і білих модулів, що утворюють горизонтальну та вертикальну лінії, які проходять між шаблонами визначення положення. Ці шаблони дозволяють визначити щільність модуляції та використовуються для правильної побудови сітки.

Область даних і кодів виправлення помилок – основна частина QR-коду, що містить закодовану інформацію (текст, URL, контакти тощо) та слова виправлення помилок, які забезпечують можливість відновлення інформації при частковому пошкодженні коду.

Інформація про версію (Version Information) – визначає номер версії QR-коду (від 1 до 40), що впливає на розмір коду та кількість модулів. Ця інформація вбудовується у вигляді специфічного шаблону поблизу шаблонів вирівнювання у великих версіях.

Інформація про формат (Format Information) – містить дані про рівень виправлення помилок та маскування шаблону, яка застосовується до QR-коду для зменшення ризику утворення занадто однорідних областей, які можуть бути складними для сканування.

Роздільники шаблонів визначення позиції – це білі рамки 8x8 модулів навколо кожного шаблону положення, що забезпечують візуальне відділення шаблону від основної області кодування.

Процес створення та розпізнавання QR-коду включає кілька чітко визначених фаз, які забезпечують правильну передачу та відновлення інформації. Ці фази можна поділити на дві основні частини: кодування (створення) та декодування (розпізнавання) як показано на рис.1.4.

Фази кодування [5]:

– першим кроком є створення або вибір повідомлення, яке має бути зашифроване в QR-коді. Це може бути будь-яка текстова інформація, наприклад, URL-адреса, контактні дані або інші дані, що потребують збереження;

– другий крок – обране повідомлення перетворюється у байтовий рядок. Це означає, що текстовий вхід (наприклад, посилання на вебсайт або номер телефону) переводиться в кодовану форму, зрозумілу для QR-коду. Кожен символ тексту кодується у вигляді байтів;

– на третьому коді – для кожного QR-коду вбудовуються дані про версію (розмір сітки QR-коду) та формат (рівень виправлення помилок і тип маскування). Ця інформація допомагає сканеру правильно інтерпретувати код під час його

сканування;

– на четвертому кроці – QR-коди використовують механізм виправлення помилок для відновлення частково пошкоджених або нечітких даних. В залежності від рівня виправлення помилок (низький, середній, квартильний або високий), генерується додатковий код, що дозволяє коригувати помилки, якщо QR-код частково втратить свою цілісність через зношення або пошкодження;

– останній крок кодування – це перетворення байтового рядка з усіма додатковими даними (виправлення помилок, інформація про версію та формат) у графічний шаблон. Цей шаблон є основною частиною QR-коду, що складається з чорних і білих модулів, що визначають закодовані дані.

Рисунок 1.4 – Процес кодування та декодування QR-коду

Фази декодування:

– крок 6 – під час декодування користувач використовує мобільний пристрій

або спеціалізований сканер для зчитування QR-коду. Камера пристрою захоплює зображення QR-коду, після чого програма сканера починає обробку;

– крок 7 – після того як QR-код було зафіксовано, сканер використовує вбудовані алгоритми для реконструкції оригінального байтового рядка. Якщо частина даних була пошкоджена, код виправлення помилок дозволяє відновити втрачену інформацію, забезпечуючи точність розпізнавання;

– крок 8 – використовуючи інформацію про версію та формат, програма сканера інтерпретує кожне кодове слово та перетворює байтовий рядок назад у вихідне повідомлення. Це може бути текст, URL, контактна інформація або інші дані, які користувач може використовувати для подальших дій.

1.2.1 Технології зчитування штрих-кодів

Технології зчитування штрих-кодів значно еволюціонували за останні десятиліття, що дозволяє використовувати їх у різних умовах і для різних типів штрих-кодів. Сучасні технології включають використання лазерних сканерів, камер і спеціалізованих датчиків.

Лазерні сканери є традиційним методом зчитування штрих-кодів. Вони працюють за принципом лазерного випромінювання, яке відбивається від штрих-коду, а спеціальний датчик фіксує зміни в інтенсивності відбитого сигналу. Лазерні сканери є дуже швидкими та точними, однак мають обмеження по куту огляду і вимогам до якості штрих-кодів.

CCD-сенсори (Charge-Coupled Device) – ці пристрої використовують кілька малих світлочутливих сенсорів, що працюють по принципу збору світлових імпульсів. CCD-сканери відрізняються високою точністю і можливістю зчитування штрих-кодів на невеликій відстані.

Інфрачервоні та світлодіодні зчитувачі – використовують інфрачервоні або світлодіодні джерела світла для сканування, що забезпечує зчитування навіть при низькому рівні освітленості.

Інтелектуальні камери – нові технології зчитування штрих-кодів включають

використання цифрових камер та інтелектуальних систем обробки зображень, які здатні здійснювати розпізнавання штрих-кодів в умовах складного фону або при деформації зображення

Камери, оснащені спеціальними алгоритмами для розпізнавання та декодування штрих-кодів, стали популярними в сучасних системах зчитування. Вони дають можливість здійснювати зчитування штрих-кодів без прямої видимості, працювати за умов поганого освітлення, а також використовувати різноманітні типи штрих-кодів (як 1D, так і 2D). Вони також здатні працювати на великих відстанях і з великими об’ємами даних. Камери можуть застосовуватися в комбінації з алгоритмами комп’ютерного зору для покращення точності зчитування.

Інфрачервоні (IR) датчики використовуються для виявлення інформації з етикеток, коли штрих-коди знаходяться в умовах низької видимості або в забруднених середовищах. З іншого боку, технологія радіочастотної ідентифікації (RFID) дозволяє автоматично зчитувати інформацію з чіпів, вбудованих в етикетки. RFID не потребує прямого контакту з чіпом або його видимості, що робить її зручним рішенням для великих складів та виробничих ліній [6].

1.2.2 Пристрої зчитування штрих-кодів

Одними з найбільш широко використовуваних пристроїв для зчитування штрих-кодів є сканери. Вони можуть мати різні конструктивні особливості та функціональні можливості.

Ручні сканери (рис.1.5) – це найбільш мобільні пристрої, які часто використовуються в торгівлі, складських приміщеннях та на складах. Вони можуть бути автономними (живлення від батарей) або підключеними до електромережі. Ручні сканери зазвичай використовуються для зчитування штрих-кодів на товарних одиницях або документах. Один з простих варіантів ручного сканера — це считывальний олівець, який здійснює сканування контактом.

Стаціонарні сканери – ці пристрої призначені для фіксації штрих-кодів без потреби переміщати товар або предмет. Стаціонарні сканери широко

використовуються на касах, в складі або на точках розрахунку. Один з варіантів таких пристроїв – щелевий лічильник, який дозволяє зчитувати штрих-коди з пластикових карт або товарів, що проходять через вузьку щілину між джерелом підсвічування і фотоприймачем [2].

Рисунок 1.5 – Лазерні сканери

Стіл-сканери – є більш складними пристроями, призначеними для автоматичного зчитування штрих-кодів з різних сторін предмета, не вимагаючи його попередньої орієнтації. Ці пристрої зручні в магазинах та торгових точках, де кожен товар може бути розміщений в будь-якому напрямку, а сканер самостійно визначає і зчитує код.

Спеціалізовані стаціонарні та вбудовуванні сканери – ці пристрої здатні здійснювати зчитування штрих-кодів на великій відстані (до 2,5 м) і з високою швидкістю (до 300 і більше сканованих в секунду). Вони ідеально підходять для використання в умовах великого потоку товарів, наприклад, на виробничих лініях або складських комплексах. Такі пристрої можуть бути оснащені портативними комп'ютерами, клавіатурами і дисплеями для обробки та відображення зчитаних даних.

Основне завдання будь-якого пристрою для зчитування штрих-кодів — це перетворення зображення штрих-коду в електричний сигнал, який можна обробити комп'ютером або іншим обчислювальним пристроєм. Сканери виконують це за допомогою оптичних систем, які складаються з джерела світла та фотоприймача. Світло, яке відбивається від штрих-коду, потрапляє на фотоприймач,

перетворюється в електричний сигнал, після чого цей сигнал підсилюється та передається до детектора для подальшого перетворення в цифрову інформацію (рис.1.6) [4].

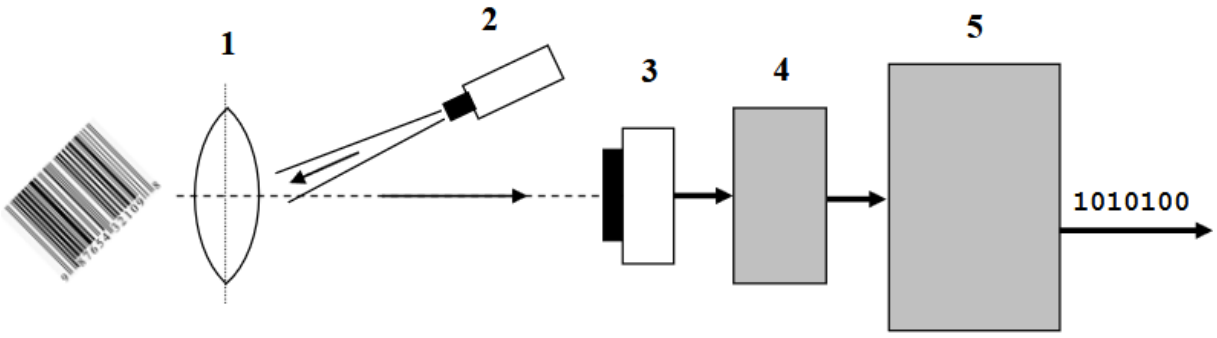

Рисунок 1.6 – Основні вузли зчитувача штрихкоду: 1 – оптична система, 2 – випромінювач світлового потоку, 3 – фотодетектор, 4 - підсилювач, 5 – детектор)

У сучасних пристроях використовуються різні оптичні сенсори: від простих світлодіодів до складних лазерних модулів і навіть цифрових камер, що дозволяють зчитувати штрих-коди з різних відстаней і під різними кутами. Зчитаний сигнал потім передається через інтерфейси сканера (USB, Bluetooth, Wi-Fi тощо) на комп'ютер або торговий термінал для подальшої обробки.

Завдяки різноманіттю пристроїв і технологій для зчитування штрих-кодів, вони стали невід’ємною частиною багатьох процесів, таких як облік товарів, управління запасами, відстеження вантажів і багато інших, забезпечуючи високу точність і ефективність у роботі підприємств різних галузей.

1.3 Огляд існуючих програмних рішень для сканування штрих-кодів

Одним з найбільш популярних підходів є використання мобільних додатків для зчитування штрих-кодів, які забезпечують зручність та мобільність у використанні. Такі додатки зазвичай працюють з вбудованими камерами мобільних телефонів для зчитування 1D та 2D штрих-кодів.

Scandit – один з найбільш відомих мобільних додатків, який дозволяє зчитувати штрих-коди за допомогою камери мобільного пристрою. Scandit забезпечує високу точність і швидкість розпізнавання навіть на пошкоджених або слабко освітлених кодах. Програмне забезпечення підтримує як 1D, так і 2D штрих-коди (QR-коди, DataMatrix тощо) і використовується в ряді бізнес-систем, таких як управління складом та автоматизація торгових процесів [7].

Barcode Scanner – додаток, який використовує камеру телефону для зчитування бар-кодів. Це просте та ефективне рішення для невеликих бізнесів або для особистого використання. Додаток підтримує багато форматів штрих-кодів, таких як UPC, EAN, QR-коди та інші [8].

Shopify POS – система, яка дозволяє бізнесам зчитувати штрих-коди товарів у роздрібних точках продажу через мобільні пристрої. Вона підтримує не тільки зчитування кодів, але й інтеграцію з базою даних для перевірки наявності товару, отримання цінової інформації та здійснення транзакцій [9].

Для великих підприємств, складів, або супермаркетів часто використовуються стаціонарні сканери, які підключаються до комп'ютера або торгового терміналу. Ці рішення здебільшого використовуються для автоматизації складських і торгових процесів, зокрема для автоматичного сканування товарів при їх обробці, відвантаженні та інвентаризації.

Zebra Technologies – один з найбільших постачальників обладнання для сканування штрих-кодів, який також пропонує програмне забезпечення для інтеграції з різними підприємницькими процесами. Це рішення дозволяє здійснювати сканування 1D і 2D кодів на великих складах, а також зчитувати коди за допомогою стаціонарних сканерів або мобільних пристроїв. Програмне забезпечення Zebra також забезпечує інтеграцію з іншими програмами для обліку, відстеження товарів і забезпечення цілісності даних [10].

Honeywell – виробник сканерів штрих-кодів, який також пропонує програмне забезпечення для підтримки операцій зчитування штрих-кодів. Вони надають рішення для автоматизації інвентаризації, відвантаження товарів та відстеження продукції через систему ERP. Програмне забезпечення підтримує інтеграцію з

великою кількістю інших систем і використовується в ряді галузей, від торгівлі до логістики [11].

Cognex – компанія, яка спеціалізується на індустріальних сканерах штрих-кодів і комп'ютерному зору. Вона надає програмні рішення для високоточних систем автоматичного зчитування штрих-кодів у складних умовах. Програмне забезпечення Cognex застосовується для обробки даних, отриманих з пристроїв, які працюють в складських і виробничих середовищах [12].

Зростаюча популярність веб-технологій привела до розвитку рішень для зчитування штрих-кодів без необхідності використання спеціалізованого програмного забезпечення. Ці рішення дозволяють зчитувати штрих-коди безпосередньо через веб-браузери, що робить їх доступними на будь-яких пристроях із камерою.

WebScan – веб-додаток, який дозволяє здійснювати сканування штрих-кодів через камеру комп'ютера або мобільного пристрою. Веб-додаток інтегрується з іншими системами через API для відправки даних до бази даних або для виконання запитів до інших програмних рішень. Це особливо зручно для магазинів, складів та підприємств, які не хочуть вкладати кошти в спеціалізоване обладнання [13].

QR Code Scanner by Online Barcode Reader – онлайн-інструмент для сканування QR-кодів, що використовує веб-камеру комп'ютера для зчитування інформації з кодів. Це швидкий і доступний спосіб отримати інформацію з QR-кодів, що може бути корисним для промоційних кампаній або мобільних додатків.

Для великих підприємств, що використовують штрих-коди в різних сферах (склади, логістика, торгівля), особливо важливою є можливість інтеграції програмного забезпечення для сканування з іншими внутрішніми системами, такими як ERP, CRM, управління ланцюгами постачань, тощо [14].

SAP Business One – це система управління підприємством, яка включає модуль для роботи зі штрих-кодами. Вона дозволяє підприємствам інтегрувати зчитування штрих-кодів в операційні процеси для автоматизації обліку товарів, інвентаризації та відвантаження [15].

Oracle NetSuite – ще одне потужне ERP-рішення, яке підтримує інтеграцію з

системами сканування штрих-кодів для управління ланцюгами постачань та складським обліком [16].

Переваги:

– більшість сучасних рішень, особливо мобільні додатки, дозволяють зчитувати штрих-коди без додаткового обладнання — лише за допомогою камери смартфона. Це робить технологію доступною навіть для малого бізнесу або особистого використання;

– онлайн-сервіси або API для сканування (наприклад, WebScan, Scandit SDK) можна легко інтегрувати у веб-додатки, що значно зменшує час на впровадження і налаштування;

– сучасні рішення підтримують більшість поширених стандартів 1D та 2D кодів, зокрема UPC, EAN, Code128, QR-коди, DataMatrix, Aztec, PDF417, тощо;

– потужні корпоративні системи, як-от SAP, Oracle NetSuite чи системи з API, дозволяють зчитані дані одразу обробляти, оновлювати інформацію про залишки на складах, генерувати звіти, тощо;

– деякі мобільні додатки дозволяють сканувати коди без підключення до Інтернету, що зручно для польових умов або віддалених складів;

– завдяки використанню звичайних камер, відсутня потреба у дорогих лазерних сканерах, що знижує поріг входу в автоматизацію.

Недоліки:

– програмні сканери (особливо ті, що працюють через камеру) можуть мати проблеми зі зчитуванням пошкоджених, забруднених або неякісно надрукованих штрих-кодів;

– при слабкому освітленні, нечіткому фокусі або тремтінні рук сканування може бути неточним або взагалі не виконатися. Це обмежує ефективність у виробничих або складських умовах;

– веб-камери або смартфони не завжди забезпечують потрібну швидкість сканування в умовах великого товарообігу (наприклад, на касі супермаркету чи в логістичному центрі);

– більшість онлайн-рішень або веб-інтерфейсів працюють лише за наявності підключення до мережі, що не завжди можливо у віддалених локаціях;

– використання сторонніх сервісів або хмарних API може становити ризики для конфіденційної інформації, особливо у фінансових, медичних або виробничих сферах;

– загальні програмні продукти можуть не враховувати специфіку певних підприємств (наприклад, сільське господарство, фармацевтика, металургія), що вимагає додаткової кастомізації.

1.4 Розробка структури веб-орієнтованної системи для сканування штрих-кодів

Об'єктом впровадження є веб-орієнтована система для сканування штрих-кодів, яка має забезпечити швидке, точне та зручне зчитування штрих-кодів різних форматів (1D, 2D), а також надання результатів у вигляді запитів до бази даних і подальшої обробки отриманих даних.

Структура, розробленої системи передбачає кілька основних компонентів:

– інтерфейс користувача (frontend) – веб-додаток, який забезпечує зручне введення даних або завантаження зображень для подальшого сканування штрих-кодів. Інтерфейс буде розроблений з урахуванням простоти та інтуїтивно зрозумілого використання, що дозволить користувачам швидко та без зайвих зусиль зчитувати та обробляти штрих-коди;

– серверна частина, яка здійснюватиме обробку запитів, розпізнавання штрих-кодів та надання результатів через API або інтерфейс. Це включає не тільки сканування та розпізнавання штрих-кодів, але й подальшу обробку отриманих даних, їх збереження та передачу до інших систем для подальшої обробки;

– база даних, що забезпечує зберігання та пошук інформації, пов’язаної з товаром, наприклад, продукція, що має штрих-код. Це також включає підтримку метаданих про товари, їх характеристиках та статусах, що дозволить виконувати пошук, фільтрацію та сортування даних;

– наявність API дозволить інтегрувати нашу систему з іншими програмними рішеннями для автоматизації логістики, складського обліку, торгових точок тощо. Це дозволяє зібрати інформацію з інших програмних середовищ та інтегрувати її в поточну систему для більшої ефективності та точності в роботі.

Особливості роботи системи включають зчитування штрих-кодів через звичайну веб-камеру, що дозволяє зменшити витрати на спеціалізоване обладнання. У реальних умовах можуть виникати проблеми з поганою видимістю штрих-кодів через бруд чи пошкодження, тому важливо передбачити алгоритми для зчитування штрих-кодів навіть за таких умов. Система повинна бути здатна працювати з різними типами штрих-кодів (1D, 2D), а також зображеннями, завантаженими користувачем.

Для забезпечення належного функціонування системи необхідно врахувати такі умови:

– система повинна працювати коректно на різних браузерах та операційних системах, забезпечуючи рівну продуктивність на всіх підтримуваних платформах;

– розпізнавання штрих-кодів повинно бути швидким і точним, включаючи можливість роботи з погано видимими або пошкодженими кодами;

– система повинна мати високу продуктивність для швидкої обробки запитів і зчитування даних, щоб забезпечити ефективність роботи в реальних умовах;

– інтерфейс повинен бути простим та зрозумілим для користувачів, навіть тих, хто не має досвіду у використанні подібних технологій.

Для реалізації серверної частини нашої системи, ми розробляємо API (Application Programming Interface), яке служить основним інтерфейсом для взаємодії клієнтської частини (frontend) з сервером. API — це набір визначень і протоколів, які дозволяють створювати та інтегрувати прикладне програмне забезпечення. У нашому випадку, для розробки API ми використовуємо NodeJS, асинхронне і кероване подіями середовище виконання JavaScript, яке спеціально призначене для створення масштабованих мережевих програм.

NodeJS надає можливість виконувати асинхронні операції, що є важливим для нашої системи, оскільки ми працюємо з великими обсягами даних, а також з численними запитами від користувачів. Завдяки цьому, система зможе обробляти запити в реальному часі з мінімальними затримками.

Для реалізації API використовуємо ExpressJS – популярний фреймворк для NodeJS, який дозволяє швидко створювати веб-додатки та API. ExpressJS забезпечує простоту в налаштуванні маршрутів, обробці запитів та управлінні сесіями.

Model-View-Controller (MVC) запропоновано використати як шаблон проектування для структурування нашої програми (рис.1.7). Модель – цей компонент відповідає за зберігання та управління даними. У нашому випадку це буде база даних, яка зберігає інформацію про товари, штрих-коди та інші пов'язані дані.

Вид – цей компонент забезпечує візуальне представлення даних для користувачів. Веб-інтерфейс є частиною компонента View, який відповідає за відображення результатів сканування.

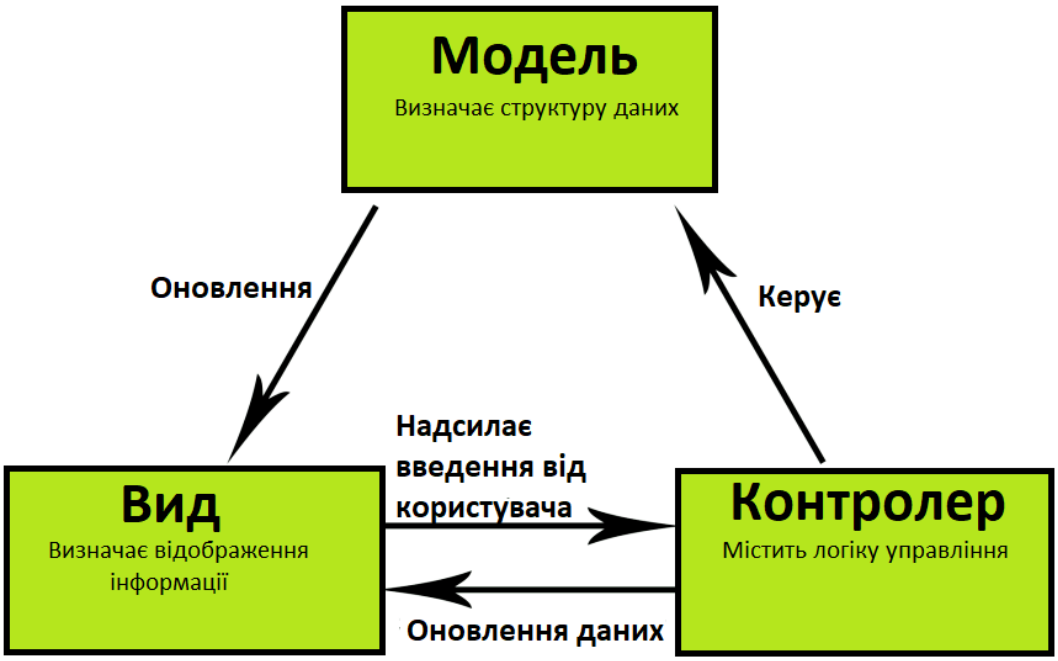

Рисунок 1.7 – Шаблон проектування «Модель-Вид-Контролер»

Контролер – цей компонент відповідає за обробку вхідних даних від

користувача, взаємодіє з моделями для отримання або оновлення інформації, а потім передає її для відображення у вигляді View.

Використання шаблону MVC дозволяє розділити бізнес-логіку від відображення та управлінської логіки, що спрощує підтримку і розширення системи.

Архітектура REST (Representational State Transfer) є основою нашого API. REST дозволяє створити ефективний і масштабований інтерфейс між клієнтом і сервером, де кожна частина може бути змінена незалежно від іншої, поки вони узгоджують формат обміну даними. Це забезпечує модульність системи та можливість її інтеграції з іншими програмами або сервісами.

Для реалізації безпечної та надійної роботи API ми використовуємо Typescript, що є надбудовою над JavaScript. Typescript забезпечує статичну типізацію, що дозволяє уникати багатьох помилок під час розробки, роблячи код більш надійним і безпечним. Використання Typescript також полегшує масштабування проекту, оскільки він дозволяє легко підтримувати великий кодовий базис без втрати продуктивності.

Щоб забезпечити безпеку нашої системи, ми використовуємо HTTPS для шифрування з'єднань між клієнтом і сервером за допомогою SSL/TLS сертифікатів. Також для аутентифікації користувачів ми впроваджуємо систему на основі OAuth, що гарантує високий рівень безпеки при авторизації та доступі до даних. Це дозволить створити стабільну, безпечну та масштабовану систему для сканування штрих-кодів, що задовольнятиме потреби сучасних бізнесів у зручному та ефективному інтерфейсі для обробки даних.

1.5 Обґрунтування вибраного напрямку вирішення задачі для веб-орієнтованної системи

UML (Unified Modeling Language) діаграма випадків використання для системи NEDS Whiteboard описує взаємодію адміністратора з різними функціями системи. Адміністратор є центральним актором, який виконує різні дії для

управління системою (рис.1.8) [17].

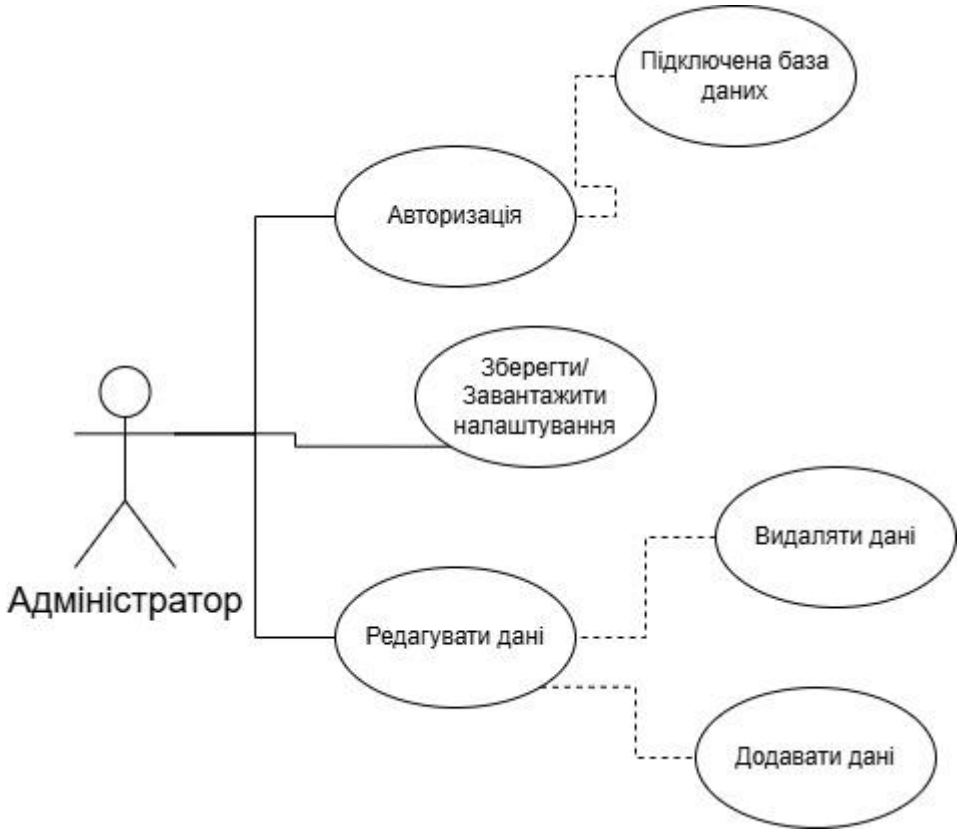

Рисунок 1.8 – UML-діаграма варіантів поведінки

Перш за все, адміністратор входить в систему, використовуючи функцію "Вхід". Після успішного входу він може завантажити або зберегти список імен за допомогою функції "Завантажити/Зберегти список імен". Якщо адміністратору потрібно отримати список імен з бази даних, він використовує функцію "Завантажити список імен з бази даних", яка взаємодіє з підключеною базою даних.

Адміністратор також має можливість додавати нові групи користувачів за допомогою функції "Додати групу". При цьому він може встановити властивості для групи, використовуючи функцію "Встановити властивості групи". Крім того, адміністратор може переміщувати або змінювати розмір групи за допомогою функції "Перемістити/Змінити розмір групи".

Для збереження та завантаження налаштувань системи адміністратор використовує функцію "Зберегти/Завантажити налаштування". Аналогічно, для

збереження та завантаження шаблонів він використовує функцію "Зберегти/Завантажити шаблон".

Адміністратор також може додавати нових людей до списку користувачів за допомогою функції "Додати людей до списку". При цьому він може переглядати список людей, використовуючи функцію "Переглянути список людей", і встановлювати властивості для окремих користувачів за допомогою функції "Встановити властивості особи". Якщо необхідно, адміністратор може видалити людей зі списку, використовуючи функцію "Видалити людей зі списку".

У UML-діаграмі на рис.1.9. представлені основні варіанти взаємодії користувачів та системи. Основні ролі включають користувача (покупця або оператора) і систему (сервіс для сканування та обробки штрих-кодів).

Користувач – людина, що взаємодіє з системою через веб-інтерфейс, завантажує або сканує штрих-код.

Система сканування – програмне забезпечення, яке обробляє запити на зчитування штрих-кодів, виконує їх обробку та видає результати.

База даних містить інформацію про товари, їх характеристики, ціни, наявність на складі тощо.

API Взаємодія з іншими бізнес-системами для передачі даних та інтеграції з іншими програмними рішеннями.

Користувач може сканувати штрих-код за допомогою камери пристрою або завантажити зображення штрих-коду.

Після сканування система перевіряє правильність і точність зчитаного штрих-коду.

Система надсилає запит до бази даних для отримання інформації про товар, таких як його назва, ціна, наявність на складі.

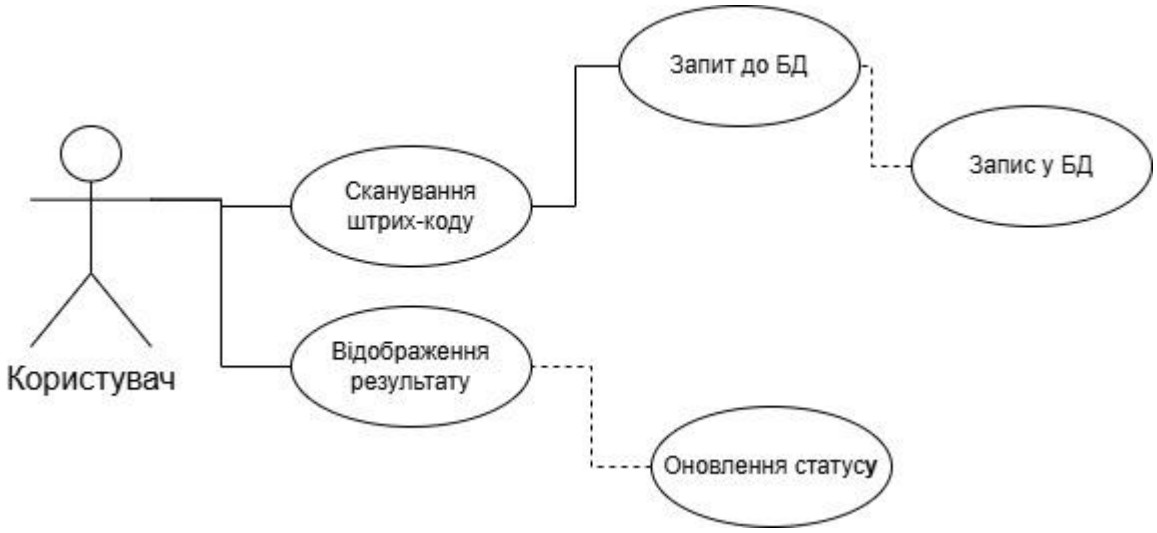

Рисунок 1.9 – UML-діаграма варіантів поведінки користувача

Система надає користувачу результат у вигляді детальної інформації про товар.

Завдяки API система може передавати дані в інші бізнес-системи для подальшої обробки, наприклад, в системи управління складом.

2 СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Технічні вимоги до веб-орієнтованної системи для сканування штрих-кодів

2.1.1 Формування загальних вимог до системи

Веб-орієнтована система для сканування штрих-кодів призначена для забезпечення швидкого, точного та доступного зчитування штрих-кодів різних форматів (1D, 2D) за допомогою веб-камери або мобільного пристрою. Основною метою системи є ідентифікація товарів та взаємодія з базою даних для отримання супровідної інформації (найменування, ціна, наявність, категорія тощо), а також можливість інтеграції з іншими програмними рішеннями в межах логістики, складського обліку або роздрібної торгівлі.

Система повинна забезпечувати зчитування штрих-кодів за допомогою звичайних веб-камер, мобільних камер або завантаженням зображень — це дозволяє обійтися без придбання дорогого спеціалізованого обладнання, значно зменшуючи вартість впровадження. Завдяки цьому користувачі можуть здійснювати сканування на будь-якому пристрої, який має доступ до Інтернету, включаючи смартфони, планшети та персональні комп’ютери.

Після зчитування штрих-коду система повинна автоматично здійснювати розпізнавання вмісту та його валідацію, далі — формувати запит до бази даних із метою отримання пов’язаної з цим кодом інформації. Це можуть бути такі характеристики товару, як: назва, опис, ціна, категорія, доступність на складі, дата останнього оновлення, постачальник, серійний номер тощо.

Ключовим аспектом системи є взаємодія з базами даних та зовнішніми програмами. Система має підтримувати API-інтерфейси для інтеграції з ERP-системами, платформами для управління складом (WMS), CRM-системами або електронною комерцією. Це дозволяє забезпечити автоматизовану передачу даних у зовнішні сервіси без додаткової участі користувача, підвищуючи загальну ефективність процесів.

У загальні вимоги до системи також входить:

– інтуїтивно зрозумілий веб-інтерфейс, який дозволяє користувачеві швидко сканувати код, переглядати результат і за потреби взаємодіяти з даними (наприклад, редагувати, експортувати або передати);

– підтримка багатокористувацького режиму з різними рівнями доступу (гостьовий режим, оператор, адміністратор);

– безпечна авторизація та автентифікація, включаючи використання JWT або OAuth2;

– можливість масштабування для обробки великої кількості запитів у разі використання системи у великих логістичних центрах чи мережевих магазинах;

– адаптивність до мобільних пристроїв: інтерфейс повинен коректно відображатися на екранах різного розміру та підтримувати сенсорне керування;

– підтримка мультимовності, принаймні українською та англійською мовами.

Додатково система повинна бути здатна працювати у реальному часі, тобто без затримок на зчитування, розпізнавання та вивід результату. Важливим є також забезпечення високого рівня надійності та захисту персональних та комерційних даних, що обробляються в процесі роботи.

2.1.2 Вимоги до структури і функціонування системи

Архітектура веб-орієнтованої системи повинна забезпечувати модульність, чіткий поділ відповідальностей між компонентами та можливість легкого розширення функціоналу. Усі функціональні блоки мають взаємодіяти між собою через стандартизовані API, а компоненти системи — бути незалежними за допомогою контейнеризації.

Система повинна складатися з таких основних компонентів:

– фронтенд (інтерфейс користувача) – веб-інтерфейс для введення, перегляду та обробки даних з можливістю сканування через камеру або завантаження зображення;

– бекенд (серверна частина) – програмна логіка для обробки запитів,

розпізнавання штрих-кодів, взаємодії з базою даних;

– база даних – зберігання інформації про товари (назва, код, категорія, ціна, наявність тощо). Структура: таблиця Products – інформація про товар. Таблиця Users – доступи адміністраторів. Таблиця ScanLogs – історія сканувань (опційно);

– API (інтерфейс прикладного програмування) – забезпечує зовнішню інтеграцію системи з іншими інформаційними системами підприємства. Запит на /api/products/:id → проксіюється до контейнера Express.js.;

– система автентифікації – забезпечує обмежений доступ для різних типів користувачів (адміністратор, оператор, гість).

Вимоги до архітектури системи:

– система повинна бути побудована на основі принципу поділу відповідальностей: окремо реалізовані фронтенд, бекенд та база даних. Це полегшує обслуговування, тестування та масштабування системи;

– усі сервіси мають бути упаковані в окремі Docker-контейнери. Контейнери повинні розгортатися у внутрішній приватній мережі Docker з обмеженим доступом, що гарантує безпечну комунікацію між сервісами;

– забезпечення автоматизованого процесу розгортання (CI/CD) із використанням GitHub Actions або іншого інструменту. Система повинна дозволяти швидке оновлення компонентів без простою;

– доступ до бази даних має бути дозволений лише з внутрішньої мережі Docker. Зовнішні запити повинні проходити через проксі-сервер (NGINX), який обробляє TLS, налаштовує HTTP-заголовки та забезпечує захист від основних мережевих атак;

– архітектура повинна передбачати горизонтальне масштабування (наприклад, запуск кількох копій API або бази даних), щоб система могла обробляти зростаючу кількість запитів.

Функціональні вимоги:

– система повинна забезпечувати зчитування штрих-кодів через веб-камеру або мобільний пристрій. Підтримуються найпоширеніші формати: EAN, UPC, QR,

DataMatrix тощо;

– інтеграція з бібліотеками розпізнавання штрих-кодів (наприклад, ZXing, QuaggaJS) для обробки зображень у браузері або на сервері;

– після зчитування штрих-коду система має здійснювати запит до бази даних, щоб отримати всю необхідну інформацію про товар: назву, ціну, наявність, категорію тощо;

– користувач повинен бачити результати сканування у зручному інтерфейсі у вигляді таблиці або картки товару. Передбачено виведення помилок у разі невдалого розпізнавання;

– відкрите REST API для взаємодії з іншими системами (ERP, CRM, WMS тощо). API повинен мати захист через токен авторизації;

– система повинна підтримувати базову рольову модель доступу: оператор (сканує товари), адміністратор (керує даними, переглядає статистику), гість (лише перегляд);

– всі дії користувачів повинні фіксуватись у журналі подій для подальшого аналізу безпеки та виявлення збоїв;

– інтерфейс повинен адаптуватися до екранів мобільних пристроїв для забезпечення зручного сканування без додаткових налаштувань.

Вимоги до способів і засобів зв'язку між компонентами системи:

– всі компоненти взаємодіють через HTTP/HTTPS-запити;

– використовується внутрішня приватна Docker-мережа з маскою /28, що забезпечує ізоляцію й безпеку;

– REST API використовує стандартні HTTP-методи: GET – отримання даних; POST – створення нових записів; PUT/PATCH – оновлення; DELETE – видалення;

– дані передаються у форматі JSON.

Передбачається взаємодія з іншими системами підприємства (ERP, CRM, складські системи), забезпечується така інтеграція (рис.2.1):

– інтерфейс: REST API;

– протокол: HTTPS;

– формат даних: JSON;

– періодичність: за потребою – запит від суміжної системи, або подія після кожного сканування.

```
{
  "product_id": "1234567890123",
  "name": "Молоко 2.5%",
  "price": 36.00,
  "quantity": 100,
  "last_updated": "2025-04-15T10:00:00Z"
}
```

Рисунок 2.1 – Фрагмент скрипта для інтеграції з іншими системами підприємства

Вимоги до режимів функціонування системи:

– основний режим – автоматизований (користувач сканує штрих-код, результат виводиться автоматично);

– режим налаштування: ручний, для адміністраторів (додавання нових товарів, керування доступом);

– цілодобова доступність (24/7) через браузер;

– відлагоджувальний режим: активується на сервері у разі потреби (через консоль Docker або API-доступ адміністратора);

– підтримка до 500 одночасних користувачів без втрати продуктивності;

– масштабування за допомогою балансування навантаження (може бути додано в майбутньому);

– база даних здатна обробляти до 1 000 000 записів товарів.

2.1.3 Вимоги до показників призначення системи

Система повинна працювати в режимі реального часу, бути доступною цілодобово з будь-якого пристрою, що має камеру та доступ до Інтернету. Вимоги

до показників призначення визначають, наскільки ефективно і стабільно система виконує свої функції згідно з її основним призначенням — зчитування штрих-кодів різних типів (1D/2D), обробка результатів та отримання інформації з бази даних. В табл. 2.1 наведено параметри описують технічні характеристики, рівень точності, швидкодії, умови експлуатації та надійності, яких повинна дотримуватися система у стандартному режимі роботи.

Таблиця 2.1 – Показники призначення веб-орієнтованої системи сканування штрих-кодів

№	Показник	Значення/Опис
1	Точність зчитування штрих-коду	Не менше 98% при нормальному освітленні (300–500 люкс)
2	Час обробки одного сканування	До 1,5 секунди
3	Час відповіді сервера (latency)	Не більше 200 мс (локально), до 500 мс (через Інтернет)
4	Мінімальна роздільна здатність камери	640×480 пікселів (VGA)
5	Пропускна здатність системи	До 500 одночасних активних сесій без втрати продуктивності
6	Рівень доступності (Uptime)	Не менше 99,5% щомісяця
7	Похибка зчитування	Не більше ±1 символ у крайніх умовах
8	Умови експлуатації	В приміщеннях класу С (захист від пилу, стабільна температура)
9	Температурний режим експлуатації	Від +10°С до +40°С
10	Вимоги до інтернет-з’єднання	Мінімальна швидкість 1 Мбіт/с
11	Підтримувані пристрої	ПК, ноутбуки, планшети, смартфони з браузерами Chrome, Firefox, Safari, Edge

2.2 Розробка апаратної частини

2.2.1 Архітектура веб-орієнтованої системи для сканування штрих-кодів

Апаратна частина веб-орієнтованої системи для сканування штрих-кодів реалізована у вигляді клієнт-серверної архітектури, що забезпечує обробку запитів користувачів, збереження та обробку даних, а також зворотний зв’язок із клієнтським інтерфейсом (рис.2.2).

Серверна логіка реалізована у вигляді REST API, створеного за допомогою платформи ExpressJS на мові програмування TypeScript. Архітектура побудована за шаблоном MVC (Model-View-Controller), що дозволяє легко масштабувати та підтримувати систему в майбутньому. Такий підхід дозволяє чітко розмежувати відповідальність між компонентами, забезпечити повторне використання коду та гнучкість при розширенні функціональності системи.

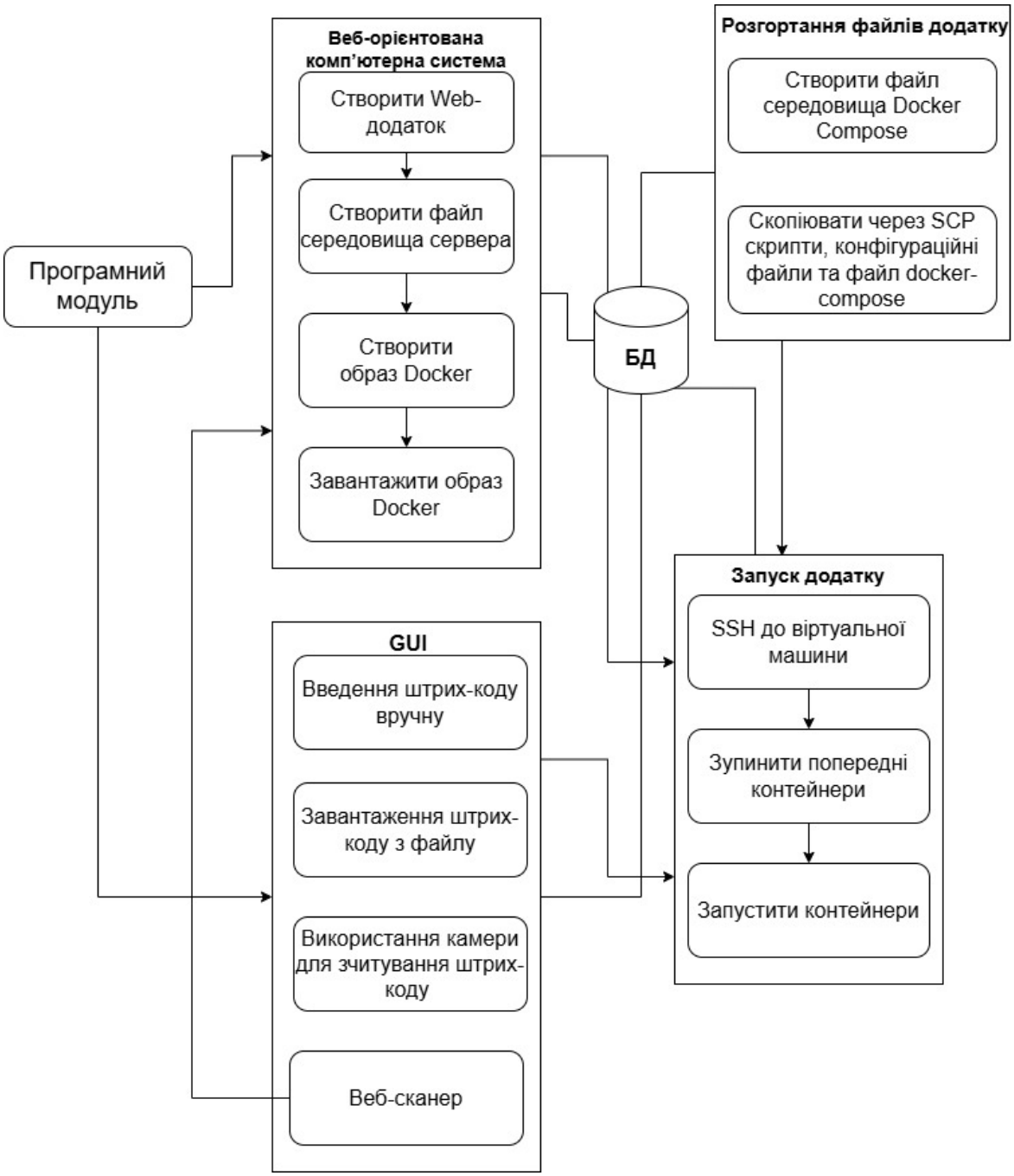

Рисунок 2.2 – Структурна схема клієнт-серверної архітектури для сканування штрих-кодів

Для збереження та обробки даних використовується система керування базами даних PostgreSQL [18] – об'єктно-реляційна СКБД з відкритим кодом. PostgreSQL було обрано завдяки її стабільності, високій продуктивності та активному розвитку. Вона підтримує складні SQL-запити, транзакції, індексацію та широкі можливості з безпеки. У рамках системи використовуються таблиці для зберігання інформації про товари, користувачів, історію сканування та логів доступу.

База даних розгортається в окремому контейнері Docker, доступ до якого можливий лише з внутрішньої мережі Docker, створеної спеціально для взаємодії компонентів системи.

Клієнтська частина реалізована на базі React.js – популярної JavaScript-бібліотеки для створення динамічних інтерфейсів. Обрано React завдяки його продуктивності, широкій екосистемі та підтримці TypeScript, що забезпечує додаткову надійність і контроль типів. Інтерфейс дозволяє:

– сканувати штрих-коди за допомогою веб-камери або мобільного пристрою.

– переглядати результати сканування та супутню інформацію про товар.

– виконувати ручний пошук або завантаження зображень штрих-кодів.

– працювати в реальному часі завдяки реактивним компонентам.

Клієнт взаємодіє з API-сервером через захищені HTTP-запити (HTTPS), що дозволяє забезпечити безпечну передачу даних.

Розгортання системи відбувається у хмарному середовищі, наприклад, Azure, з використанням інструментів автоматизації CI/CD. Основні етапи розгортання включають:

– Build APP – створення Docker-образу додатку та надсилання його до репозиторію Docker Hub;

– Deploy App Files – передача лише необхідних конфігураційних файлів, Dockerfile та скриптів розгортання на сервер;

– розгортання через SSH – запуск серії команд оболонки, які перезапускають сервіси на сервері без потреби копіювати весь репозиторій.

Контейнери працюють у приватній віртуальній мережі з підмережею /28. Це

обмежує кількість доступних IP-адрес, забезпечує безпеку та ізоляцію середовища. База даних приймає з'єднання лише з IP-адрес цієї мережі, що запобігає несанкціонованому доступу ззовні.

Як показано на рис.2.3, веб-сервер NGINX забезпечує TLS для з’єднань, завантажуючи сертифікати SSL і додаючи заголовки безпеки до всіх HTTP-відповідей. Це зміцнює зв’язок між клієнтом і сервером, гарантуючи зашифрований канал зв’язку, видаляючи небажані HTTP-заголовки, які можуть розкривати зайву інформацію, а також дозволяючи лише необхідні методи HTTP, відхиляючи потенційно небезпечні, наприклад, метод HTTP OPTION.

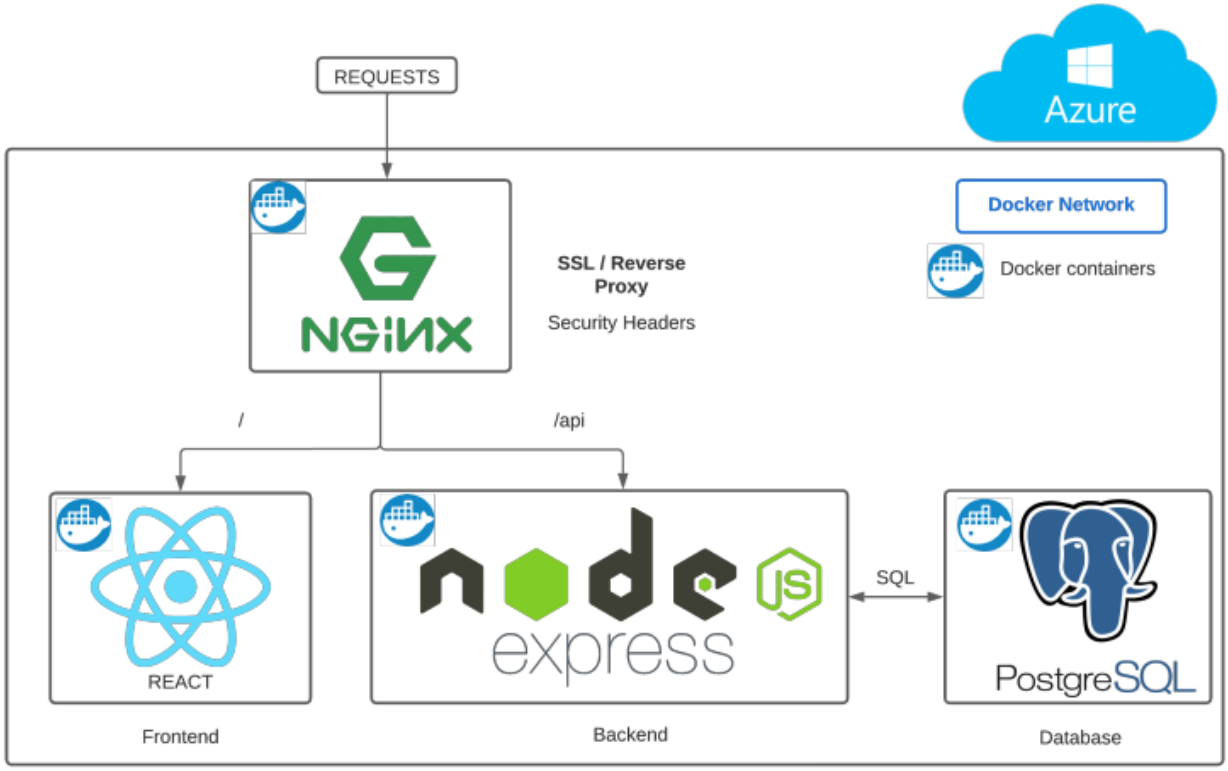

Рисунок 2.3 – Структурна схема веб-орієнтована комп'ютерна система

Вся конфігурація TLS та HTTP заголовків здійснюється на рівні веб-сервера, а не на сервері API, оскільки веб-сервер відповідає за абстрагування цих налаштувань, щоб сервер міг зосередитись виключно на виконанні бізнес-логіки. Зворотна проксі-конфігурація дозволяє перенаправляти вхідні запити до відповідних сервісів. Наприклад, якщо запит стосується ресурсу /api, він буде перенаправлений на контейнер Express-сервера, який обробить запит, виконає операції з базою даних та

надасть відповідь. Водночас, якщо клієнт звертається до домашньої сторінки (/), NGINX повертає клієнту веб-сторінку.

Основний сервер побудований на REST API, реалізованому за допомогою ExpressJS і написаному на TypeScript.

Для збереження даних вибрано PostgreSQL — потужну об’єктно-реляційну систему керування базами даних з відкритим кодом, яка вирізняється надійністю та високою продуктивністю. Обрання PostgreSQL обумовлене її популярністю у галузі та підтримкою SQL стандартів, що робить її ідеальним вибором для цієї системи.

Фронтенд-частина створена за допомогою React.js, популярної бібліотеки JavaScript для побудови інтерфейсів користувача. React був обраний завдяки активній спільноті розробників, великій кількості доступних пакетів, а також підтримці TypeScript, що дозволяє забезпечити строгий контроль типів і підвищити надійність коду. React дозволяє легко інтегрувати лише необхідні пакети, що дозволяє уникнути зайвих залежностей і забезпечує більшу гнучкість при розробці інтерфейсу користувача.

2.3 Програмування веб-орієнтованої системи для сканування штрих-кодів

2.3.1 Призначення програми

Розроблена програма призначена для реалізації функціональності сканування штрих-кодів та обробки відповідної інформації про товар у складі веб-орієнтованої облікової системи. Програмне забезпечення входить до складу інформаційного комплексу, який забезпечує автоматизацію обліку товарів та спрощення процесу ідентифікації продукції в торгівельних або складських об'єктах. Основна функціональна зона – це блок первинного збору, обробки та виводу інформації про товарну одиницю, тобто фронт-енд частина складу або точки продажу, де виконується швидке зчитування штрих-коду та подальші операції.

На рис. 2.4 наведено схему алгоритму роботи програмного забезпечення веб-орієнтованої системи сканування штрих-кодів наочно ілюструє основні етапи обробки даних та взаємодії між різними компонентами системи

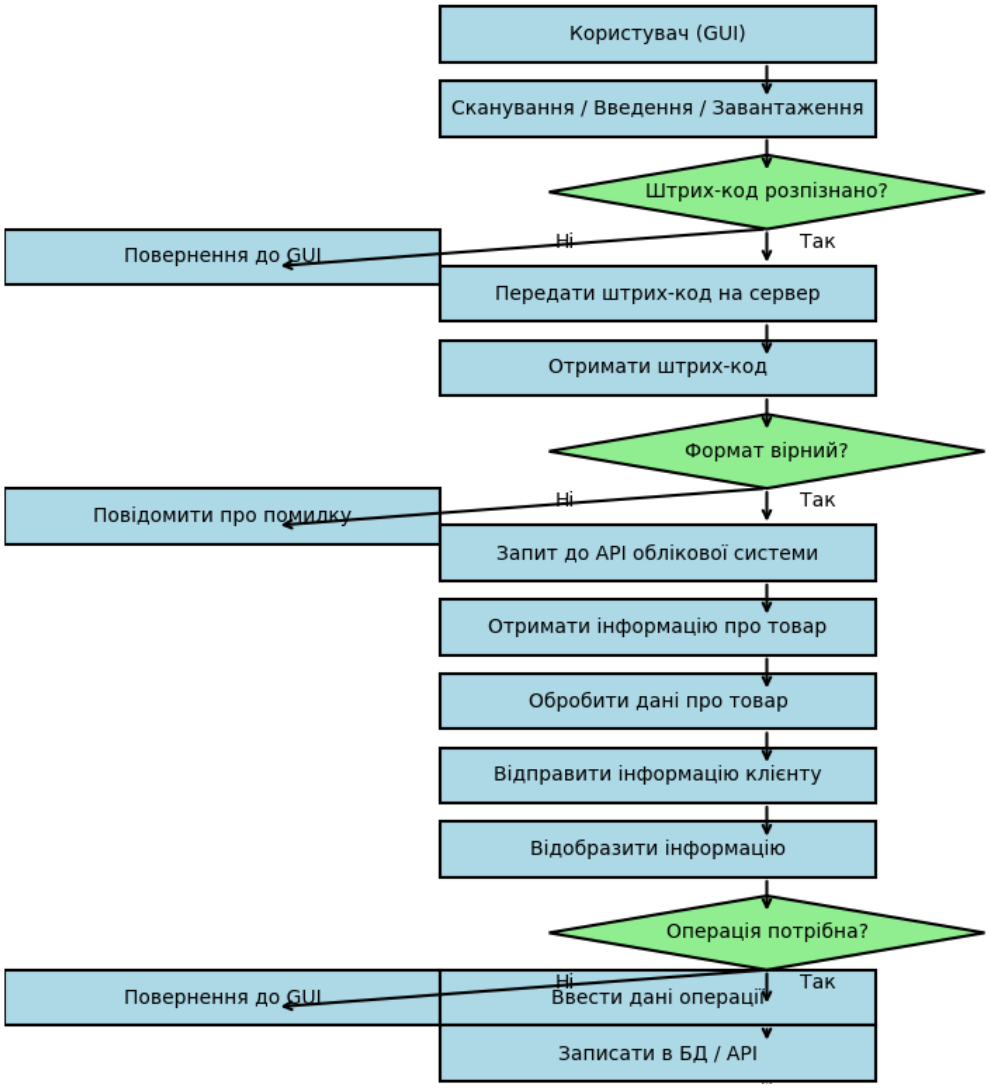

Рисунок 2.4 – Структурна схема алгоритму роботи програмного забезпечення веб-орієнтованої системи

Першим етапом є взаємодія користувача з клієнтською частиною, яка представлена веб-браузером. Співробітник складу ініціює процес ідентифікації товару шляхом введення штрих-коду вручну або сканування штрих-коду за допомогою камери мобільного пристрою чи підключеного сканера. Вхідними даними на цьому етапі є зображення штрих-коду (отримане з камери або файлу) або текстовий рядок штрих-коду (введений користувачем). Після успішного зчитування та розпізнавання штрих-коду клієнтська частина формує запит, що містить текстовий рядок штрих-коду та його формат, і передає його на сервер. Ці дані є вихідними даними клієнтської частини, призначеними для серверної обробки.

Другим етапом є обробка отриманих даних серверною частиною, яка реалізована на базі Node.js з використанням фреймворку Express.js. Сервер отримує вхідні дані у вигляді тексту штрих-коду та його формату. Першим кроком сервер здійснює валідацію формату штрих-коду, щоб переконатися у його відповідності підтримуваним стандартам. У разі успішної валідації сервер формує та надсилає запит до API облікової системи. Вхідними даними для цього запиту є текст штрих-коду. У випадку, якщо формат штрих-коду є невірним, сервер формує та відправляє повідомлення про помилку назад клієнту. Це повідомлення є вихідними даними серверної частини, призначеними для відображення користувачеві.

Третім етапом є взаємодія з API облікової системи, яка являє собою зовнішню систему, що містить актуальну інформацію про товари. API облікової системи отримує вхідні дані у вигляді тексту штрих-коду та здійснює пошук відповідної інформації про товар у своїй базі даних. У результаті обробки запиту API облікової системи повертає вихідні дані на сервер у вигляді інформації про товар (назва, ціна, кількість, тощо).

Четвертим етапом знову є робота серверної частини. Сервер отримує вхідні дані у вигляді інформації про товар від API облікової системи. Він обробляє ці дані та формує відповідь для клієнтської частини, яка містить необхідну для відображення інформацію. Вихідними даними серверної частини на цьому етапі є інформація про товар, яка передається назад у веб-браузер користувача.

П'ятим етапом є відображення отриманої інформації в клієнтській частині. Веб-браузер отримує вхідні дані у вигляді інформації про товар та відображає їх користувачеві у зручному графічному інтерфейсі. Користувач переглядає отримані дані та може ініціювати подальші дії, наприклад, фіксацію складської операції. Для цього користувач обирає тип операції (приймання, відвантаження тощо) та вводить необхідні параметри (наприклад, кількість товару). Ці дані про обрану операцію та її параметри стають вхідними даними для наступного етапу.

Шостим етапом є передача даних про складську операцію з клієнтської частини на серверну частину. Вихідними даними клієнтської частини є дані про обрану складську операцію та її параметри, які надсилаються на сервер.

Сьомим етапом є обробка даних про операцію сервером. Сервер отримує вхідні дані про складську операцію та її параметри. Він здійснює валідацію отриманих даних, а потім, за необхідності, записує інформацію про операцію до локальної бази даних (наприклад, для ведення локального журналу операцій). Після цього сервер формує запит та передає дані про складську операцію до API облікової системи для оновлення облікових даних. Ці дані є вихідними даними серверної частини, призначеними для API облікової системи.

Восьмим етапом є обробка запиту на фіксацію операції API облікової системи. API облікової системи отримує вхідні дані про складську операцію та здійснює відповідні зміни у своїй базі даних. У результаті обробки API облікової системи повертає вихідні дані на сервер у вигляді статусу виконання операції (успішно або з помилкою).

На завершальному, дев'ятому етапі, серверна частина отримує вхідні дані про статус виконання операції від API облікової системи та формує відповідне повідомлення для клієнтської частини. Це повідомлення, що відображає статус виконання операції (успіх або помилка), є вихідними даними серверної частини, які передаються назад у веб-браузер користувача для інформування про результат виконаної дії.

2.3.2 Обґрунтування технічних характеристик

Розроблене програмне забезпечення вирішує низку задач, спрямованих на автоматизацію процесу сканування товарів та обробки отриманої інформації. Ось основні функціональні завдання, які вирішуються за допомогою програми:

– користувач може здійснити сканування товару за допомогою камери, або вручну ввести штрих-код, якщо сканер не доступний або не працює належним чином;

– після введення штрих-коду програма надає користувачеві візуальне представлення інформації про товар, таку як його назва, ціна та інші параметри;

– програма підтримує завантаження зображень, що містять штрих-коди, і

їх подальше розпізнавання. Це дозволяє користувачеві працювати не тільки з камерою, а й із збереженими файлами;

– усі операції з товарами (наприклад, додавання нових товарів або оновлення кількості на складі) зберігаються в локальній базі даних. Це дозволяє забезпечити надійне зберігання та швидкий доступ до необхідної інформації;

– програма виводить текстову інформацію про товар, зокрема його назву, ціну та кількість, що допомагає користувачам швидко ознайомлюватися з характеристиками товару після сканування або введення штрих-коду;

– додавання товару до "кошика" та зменшення кількості на складі. Користувач може додавати товар до "кошика" або виконувати інші операції, які змінюють кількість товару в базі даних (наприклад, покупка товару);

– програма підтримує інтеграцію з веб-сканерами через браузер, що дає можливість користувачам використовувати сторонні інструменти для сканування штрих-кодів.

Програма працює з кількома типами вхідних даних:

– отримується шляхом сканування штрих-коду за допомогою камери або через введення вручну;

– якщо товар не має штрих-коду або є потреба в завантаженні додаткової інформації, можна завантажити зображення, що містить штрих-код;

– кількість одиниць товару для додавання до кошика: це дані, які користувач вводить після вибору товару для здійснення покупки чи іншої операції.

Після обробки вхідних даних програма надає наступну інформацію:

– назва товару, ціна, кількість на складі. Ці дані виводяться на екран і допомагають користувачеві здійснити подальші операції;

– у разі невдалого сканування або інших помилок програма виводить відповідні повідомлення для користувача;

– після успішного додавання товару до кошика або оновлення даних користувач отримує підтвердження.

Для розробки даного програмного забезпечення був обраний Python як мова програмування. Tkinter було вибрано для побудови графічного інтерфейсу завдяки своїй простоті у використанні та вбудованості в Python. Tkinter дозволяє швидко створювати інтерфейси з кнопками, полями введення та іншими елементами для взаємодії з користувачем, що робить програму зручною і зрозумілою для кінцевого користувача.

SQLite вибрано для збереження та обробки даних через його легкість у інтеграції та відсутність необхідності в окремих серверних рішеннях. SQLite – це вбудована база даних, яка не потребує налаштування сервера, а також забезпечує швидку обробку невеликих обсягів даних, що ідеально підходить для даної програми [19].

Для обробки зображень та сканування штрих-кодів було використано бібліотеки PIL для завантаження та обробки зображень, а також OpenCV для роботи з відеопотоком, що дозволяє здійснювати реальний моніторинг та зчитування штрих-кодів без необхідності в додаткових пристроях.

Обраний склад технічних і програмних засобів забезпечує високу продуктивність програми, її зручність для користувача, а також можливість легкої інтеграції з іншими системами або програмами у майбутньому.

2.3.3 Опис розробленої програми

Графічний інтерфейс користувача реалізовано з використанням стандартної бібліотеки Tkinter [20], яка є частиною мови програмування Python (рис.2.5). Основне вікно програми забезпечує користувачу доступ до всіх функцій сканера: ручного введення штрих-коду, завантаження зображення з кодом, активації камери для отримання відеопотоку, доступу до онлайн-сканера та перегляду інформації про товар. Кожна функціональна кнопка супроводжується відповідною дією, а результати виводяться у вигляді текстового повідомлення під головним меню. Інтерфейс також містить окреме поле для виводу відеопотоку з камери, яке динамічно оновлюється. Усі елементи інтерфейсу згруповані у функціональні блоки, що дозволяє користувачу інтуїтивно взаємодіяти з програмою.

```
self.input_button = tk.Button(self.root, text="Ввести штрих-код вручну", command=self.manual_input)
self.result_display = tk.Label(self.root, text="", font=("Arial", 14, "bold"), fg="green", justify="left")
```

Рисунок 2.5 – Фрагмент коду реалізації графічного інтерфейсу

Розпізнавання штрих-кодів у програмі здійснюється наразі в напівручному режимі. Користувач може або самостійно ввести код з етикетки, або завантажити зображення товару, після чого програма пропонує ввести зчитаний код вручну. У подальших версіях передбачено інтеграцію бібліотеки pyzbar для автоматичного розпізнавання штрих-кодів з камери або фото (рис.2.6). Після введення коду система звертається до бази даних і на основі введеного штрих-коду отримує відповідну інформацію про товар: його назву, ціну та кількість на складі. Якщо товар знайдено, дані виводяться у вигляді структурованого тексту у спеціально відведеному полі. У разі відсутності товару система повідомляє користувача про це червоним повідомленням. Крім того, користувач може додати товар у кошик, вказавши кількість, після чого система зменшує відповідний залишок у базі даних.

```
from pyzbar.pyzbar import decode
from PIL import Image

def update_frame(self):
    if not self.cap:
        return
    ret, frame = self.cap.read()
    if ret:
        try:
            # Перетворення кадру OpenCV у зображення PIL
            frame_pil = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
            # Розпізнавання штрих-кодів
            barcodes = decode(frame_pil)
            for barcode in barcodes:
                barcode_data = barcode.data.decode('utf-8')
                print(f"Знайдено штрих-код: {barcode_data}")
                self.check_and_display_product(barcode_data)
            except Exception as e:
            print(f"Помилка розпізнавання: {e}")

        frame_tk = ImageTk.PhotoImage(frame_pil)
        self.camera_frame.config(image=frame_tk)
        self.camera_frame.image = frame_tk
    self.root.after(10, self.update_frame)
```

Рисунок 2.6 – Фрагмент коду реалізації алгоритму штрих-кодів

База даних реалізована на основі вбудованої СУБД SQLite, що дозволяє зберігати інформацію про товари локально, без потреби у серверному рішенні. Структура бази даних містить таблиці товари, операції, користувачи, логи у якій зберігається інформація про штрих-код, назву товару, ціну та кількість. Основні

операції з базою даних включають: додавання товарів, отримання інформації про товар за штрих-кодом, перевірку наявності записів у таблиці (рис.2.7), а також зменшення кількості товарів при додаванні до кошика. Усі запити реалізовані за допомогою бібліотеки sqlite3, а з’єднання з БД ініціалізується при старті програми.

Таблиця "Товари" є центральною сутністю системи та призначена для зберігання детальної інформації про кожен товар, що обліковується на складі. Вона включає наступні атрибути: унікальний ідентифікатор товару (id типу INTEGER, що є первинним ключем), унікальний штрих-код товару (barcode типу TEXT з обмеженням UNIQUE), назву товару (name типу TEXT), ціну товару (price типу REAL), кількість товару на складі (quantity типу INTEGER) та опис товару (description типу TEXT). Первинний ключ забезпечує однозначну ідентифікацію кожного товару, а унікальний штрих-код дозволяє швидко та безпомилково ідентифікувати товар під час сканування або введення.

Таблиця "Операції" призначена для фіксації всіх транзакцій, що відбуваються з товарами на складі. Кожен запис у цій таблиці відображає окрему операцію та містить таку інформацію: унікальний ідентифікатор операції (id типу INTEGER, що є первинним ключем), ідентифікатор товару, з яким пов'язана операція (product_id типу INTEGER, що є зовнішнім ключем, який посилається на поле id таблиці "Товари"), тип операції (operation_type типу TEXT, що вказує на характер дії, наприклад, приймання, відвантаження), кількість товару, залученого в операції (quantity типу INTEGER), та час виконання операції (timestamp типу DATETIME). Зовнішній ключ product_id забезпечує цілісність даних та зв'язок між операціями та конкретними товарами.

Таблиця "Користувачі" зберігає облікові дані співробітників, які мають доступ до системи. Вона містить наступні поля: унікальний ідентифікатор користувача (id типу INTEGER, що є первинним ключем), унікальний логін користувача (username типу TEXT з обмеженням UNIQUE), пароль користувача (password типу TEXT) та роль користувача в системі (role типу TEXT, що визначає його права та привілеї). Унікальний логін забезпечує однозначну ідентифікацію кожного користувача в системі.

Таблиця "Логи" призначена для аудиту дій користувачів та відстеження подій у системі. Кожен запис у цій таблиці фіксує дію, виконану користувачем, та містить таку інформацію: унікальний ідентифікатор запису логу (id типу INTEGER, що є первинним ключем), ідентифікатор користувача, який виконав дію (user_id типу INTEGER, що є зовнішнім ключем, який посилається на поле id таблиці "Користувачі"), опис виконаної дії (action типу TEXT) та час виконання дії (timestamp типу DATETIME). Зовнішній ключ user_id дозволяє відстежувати, який саме користувач ініціював певну дію в системі.

Зв'язки між таблицями реалізовані за допомогою зовнішніх ключів (рис.2.6), що забезпечує цілісність даних та відображає логічні зв'язки між сутностями. Зокрема, існує зв'язок "один-до-багатьох" між таблицею "Товари" та таблицею "Операції" (один товар може мати багато пов'язаних з ним операцій), а також між таблицею "Користувачі" та таблицею "Логи" (один користувач може виконати багато дій, які фіксуються в логах).

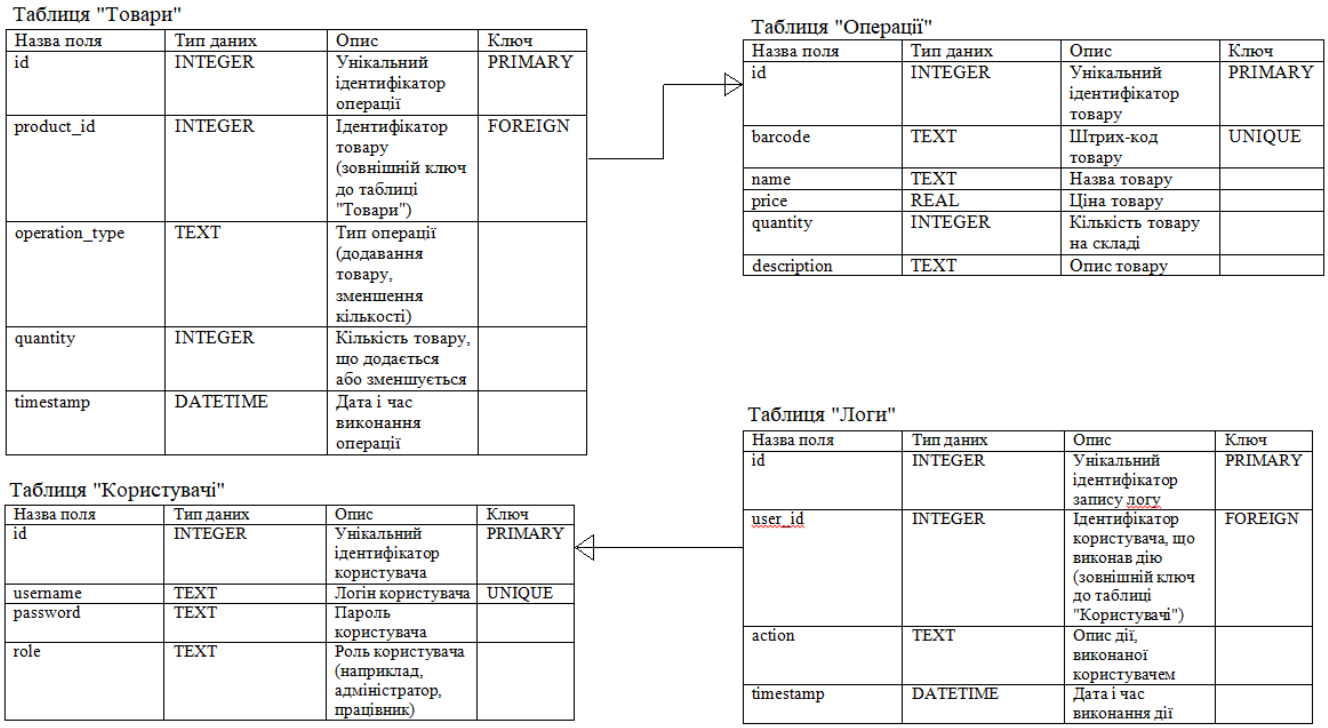

Рисунок 2.7 – Структура БД веб-орієнтованої системи для сканування штрих-кодів

Окрім десктопної частини, розроблена система також має підтримку веб-орієнтованого інтерфейсу. Для цього реалізовано можливість відкриття онлайн-сканера у браузері, який дозволяє зчитувати штрих-коди безпосередньо за допомогою камери смартфона чи комп’ютера. Веб-інтерфейс реалізований за допомогою JavaScript-бібліотеки React, що забезпечує високу продуктивність, швидке оновлення компонентів інтерфейсу та гнучкість при розширенні функціоналу. React дозволяє формувати компонентну структуру інтерфейсу, забезпечуючи повторне використання елементів, а також інтегрується з TypeScript для забезпечення кращої типізації та стабільності коду. Веб-інтерфейс взаємодіє з серверною частиною через REST API, що дозволяє централізовано обробляти запити від користувача.

2.4 Взаємодія користувача з веб-орієнтованою системою для сканування штрих-кодів

Першим кроком у взаємодії користувача з розробленою веб-орієнтованою системою сканування штрих-кодів є робота з її графічним інтерфейсом, представленим у веб-браузері. Інтерфейс розроблено з метою забезпечення інтуїтивно зрозумілої та ефективної взаємодії співробітника складу з основними функціональними можливостями системи (рис.2.8).

При натисканні на кнопку "Ввести штрих-код вручну" користувачеві надається текстове поле для введення штрих-коду з клавіатури. Це є альтернативним способом ідентифікації товару у випадках, коли сканування неможливе або недоступне.

Кнопка "Завантажити зображення" відкриває стандартний діалог вибору файлів операційної системи, дозволяючи користувачеві завантажити зображення, що містить штрих-код. Після вибору файлу система здійснює спробу розпізнати штрих-код на завантаженому зображенні.

Кнопка "Відкрити камеру" ініціює запит на доступ до камери пристрою (веб-камери комп'ютера або камери мобільного пристрою). Після надання дозволу на

екрані відображається вікно попереднього перегляду з камери, що дозволяє користувачеві навести камеру на штрих-код для його сканування в режимі реального часу.

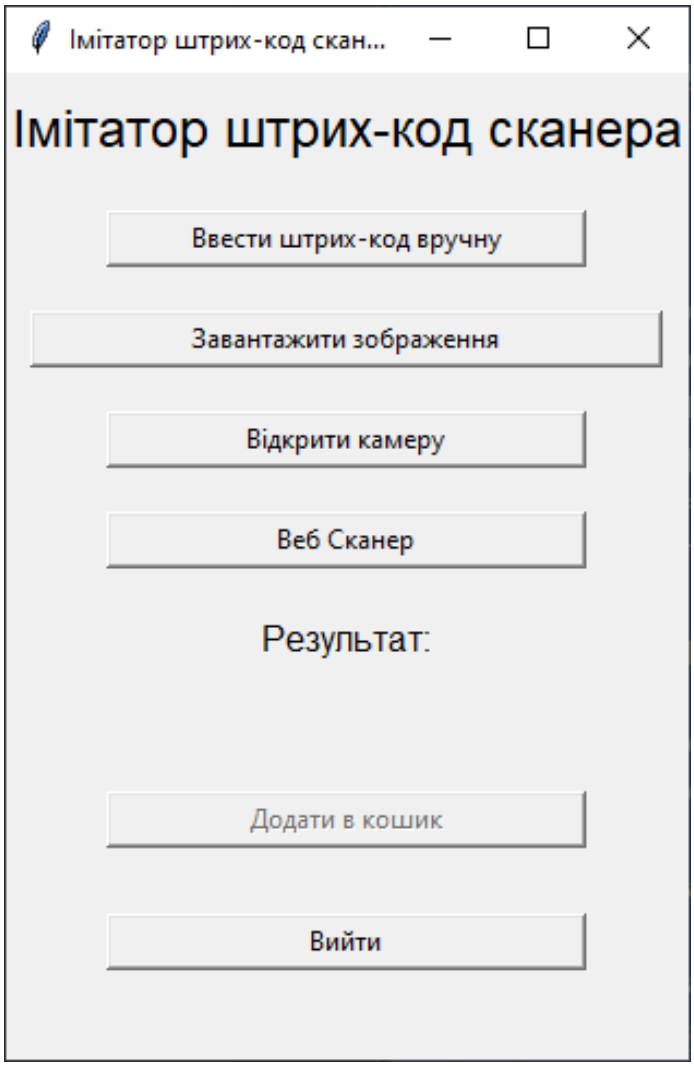

Рисунок 2.8 – Головний інтерфейс системи для сканування штрих-кодів

Кнопка "Веб Сканер" призначена для інтеграції зі стороннім веб-сервісом або плагіном веб-браузера, який виконує функцію сканування штрих-кодів.

Після первинного відображення інтерфейсу та успішної ідентифікації товару одним із доступних методів (введення вручну, завантаження зображення, сканування камерою або веб-сканером), система відображає детальну інформацію про знайдений товар у спеціальному блоці, позначеному як "Результат:". Цей блок є ключовим для надання користувачеві візуального підтвердження правильності ідентифікації та відображення основних характеристик товару.

На рис.2.9 після зчитування штрих-коду "4820066948988" у блоці "Результат:" відображається наступна інформація: безпосередньо сам штрих-код ("4820066948988"), ідентифікатор або назва товару ("Товар 63"), його ціна ("935,94 грн") та поточний залишок на складі ("339 шт"). Відображення цієї інформації дозволяє користувачеві візуально переконатися у правильності розпізнавання товару перед виконанням подальших дій. У реальній експлуатації системи поле "Товар" може містити повну назву товару для кращого розуміння користувачем.

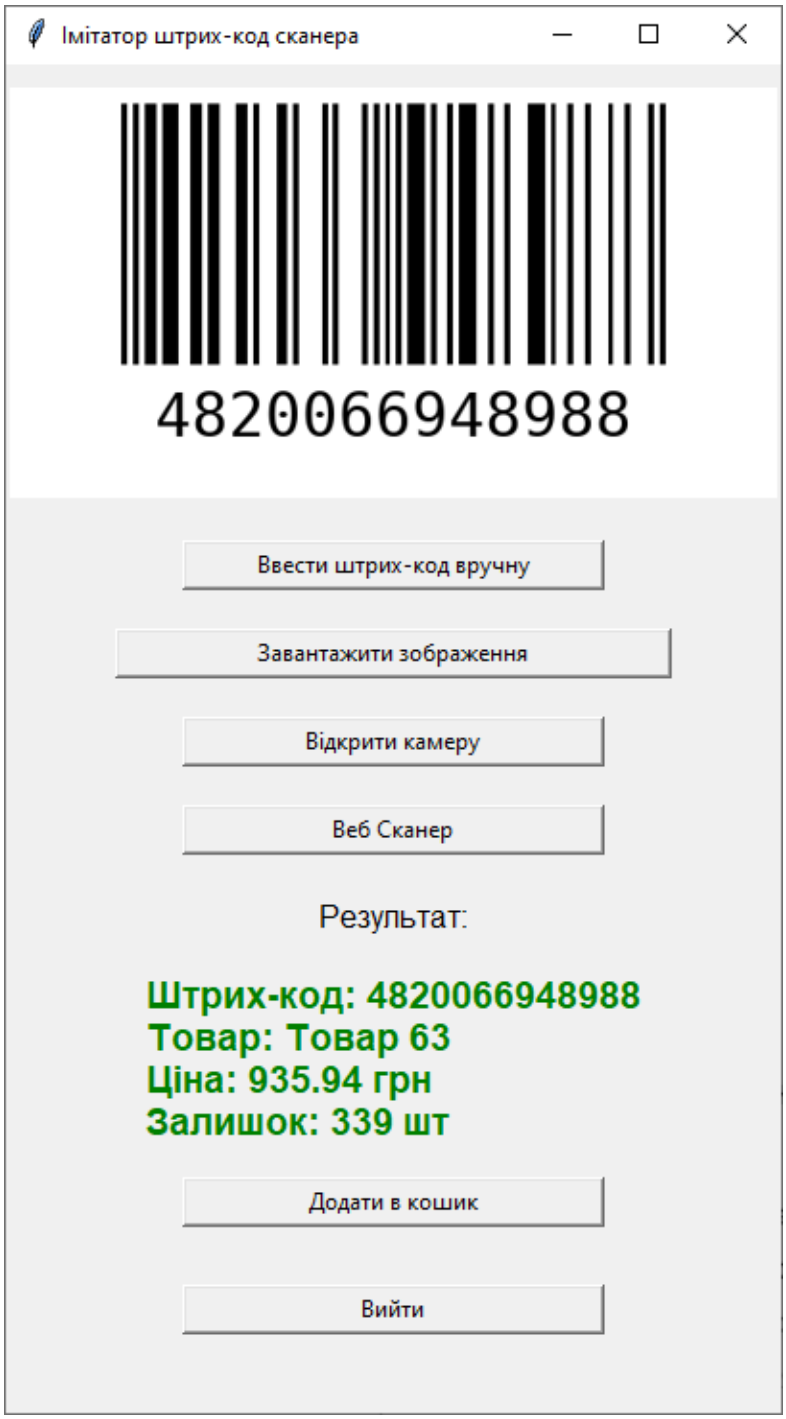

Рисунок 2.9 – Результат інформації штрих-коду

Після ознайомлення з інформацією про товар користувач отримує можливість ініціювати подальші складські операції за допомогою інтерактивних елементів керування, розташованих у нижній частині інтерфейсу. На даному етапі користувачеві доступна кнопка "Додати в кошик" (рис.2.10). Натискання цієї кнопки є сигналом для системи про намір користувача здійснити певну складську операцію з відображеним товаром. Логіка подальших дій після натискання кнопки "Додати в кошик" відображає віртуальний кошик з доданим товаром, де користувач зможе вказати необхідну кількість для операції (рис.2.11).

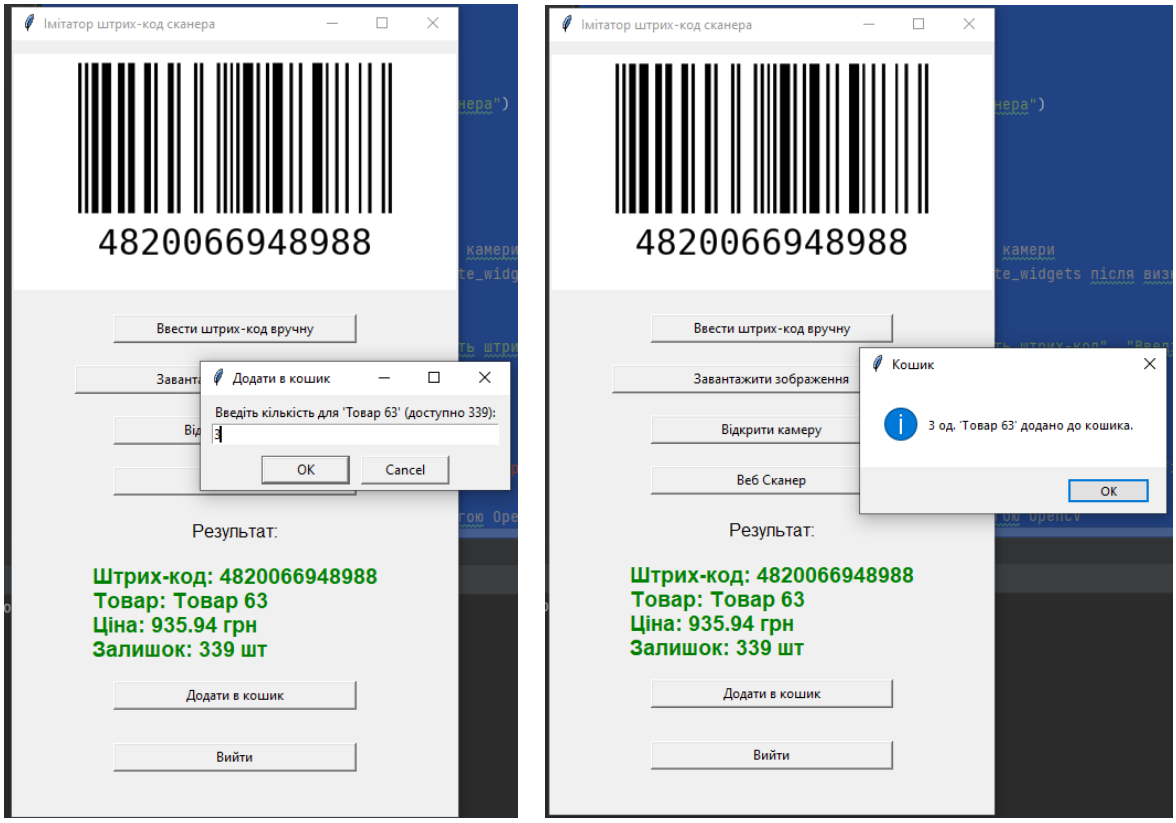

Рисунок 2.10 – Результат функціоналу «Додати в кошик»

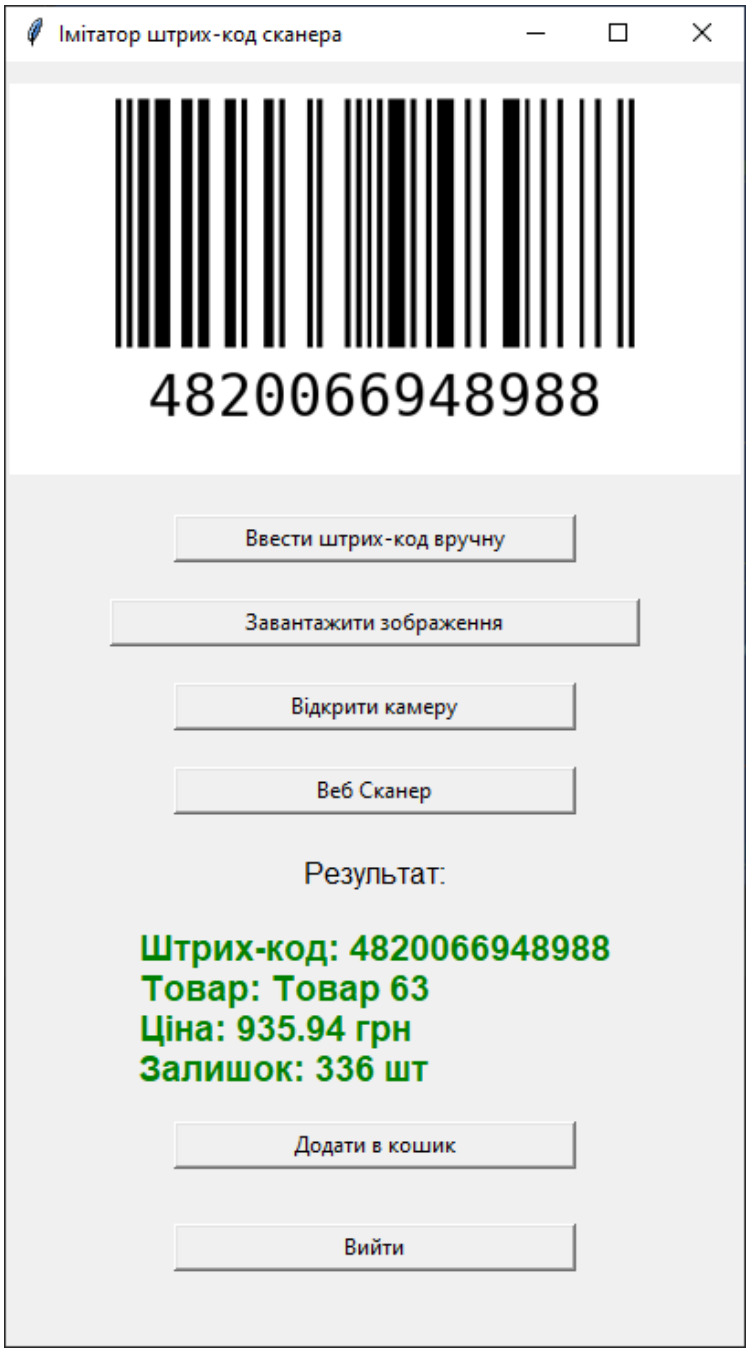

Рисунок 2.11 – Результат після додавання до кошику

В іншому випадку, система може автоматично додавати одну одиницю товару до кошика з можливістю подальшого редагування його вмісту на окремому екрані або в іншому блоці інтерфейсу.

При натисканні кнопки "Веб Сканер" користувача буде перенаправлено на URL-адресу цього сервісу (рис.2.12). На цьому сервісі користувач зможе використовувати веб-камеру свого пристрою для сканування або завантажити зображення з кодом. Після успішного сканування результат (розшифрований текст

або посилання) може бути переданий назад у основну систему або відображений безпосередньо на сторінці веб-сканера.

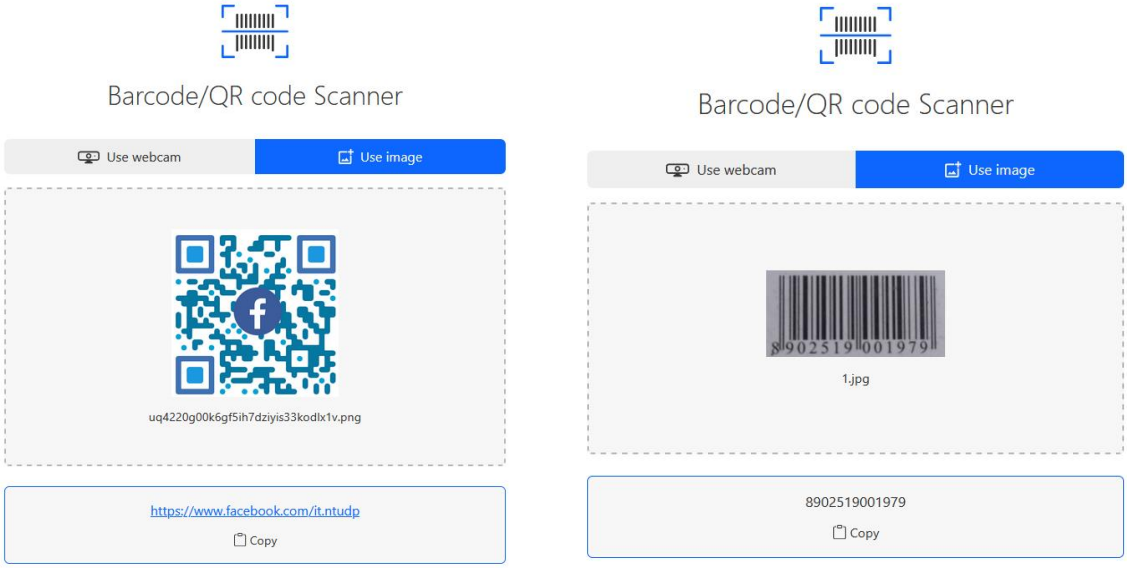

Рисунок 2.12 – Результат модуля "Веб Сканер"

Кнопка "Веб Сканер" ініціює використання WebRTC API браузера для доступу до веб-камери пристрою, наданий сторонньою бібліотекою JavaScript (рис.2.13). Цей інтерфейс може мати додаткові функції, такі як налаштування області сканування, вибір типу коду для сканування тощо.

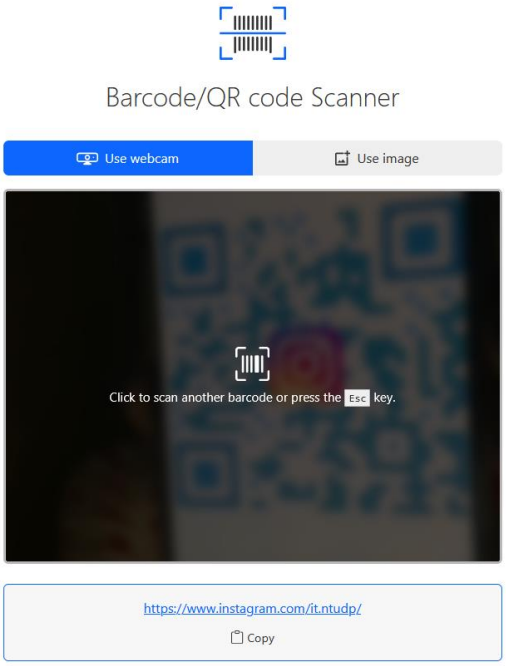

Рисунок 2.13 – Результат модуля "Веб Сканер" в режимі камери

Таким чином, після успішної ідентифікації товару інтерфейс системи сканування штрих-кодів надає користувачеві не лише детальну інформацію про товар, але й інструменти для ініціювання подальших складських операцій. Натискання кнопки "Додати в кошик" є ключовим кроком для переходу до наступних етапів обліку та управління товарами, а наявність кнопки "Вийти" забезпечує можливість безпечного завершення роботи з системою. Подальша взаємодія користувача з системою залежатиме від обраної складської операції та відповідних бізнес-процесів, реалізованих у програмному забезпеченні.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи на тему: «Веб-орієнтована комп’ютерна система з інтеграцією API для сканування штрих-кодів різних форматів» було повністю реалізовано поставлену мету – створення функціональної програмної системи, здатної розпізнавати, обробляти та відображати інформацію про товари за штрих-кодом, використовуючи локальні та зовнішні джерела даних. Програма забезпечує зручний інтерфейс для взаємодії користувача з базою товарів, підтримує ручне введення штрих-кодів, обробку зображень із файлів, використання веб-камери, а також інтеграцію зі сторонніми веб-сервісами через API.

Реалізовано багатофункціональний графічний інтерфейс, забезпечено стабільну роботу з локальною базою даних, реалізовано можливість розширення функціональності шляхом підключення веб-сервісів. Система правильно опрацьовує вхідні дані (штрих-коди у різних форматах), коректно виводить інформацію про товари, обробляє помилки та дозволяє оновлювати залишки товарів шляхом зменшення кількості при фіксації продажу. Всі функціональні блоки системи протестовано й підтверджено їх працездатність.

Узагальнюючи результати, можна стверджувати, що всі задачі, поставлені у рамках кваліфікаційної роботи, були успішно вирішені. Система забезпечує комплексне обслуговування процесу сканування та обліку товарів, демонструючи високу гнучкість у роботі з різними джерелами даних. Мета роботи – розробити ефективну, сучасну й зручну у використанні веб-орієнтовану комп’ютерну систему для ідентифікації товарів – досягнута повністю.

Практичне значення отриманих результатів полягає у можливості впровадження розробленої системи в малих та середніх торгових підприємствах, логістичних компаніях, складах, бібліотеках або в системах внутрішнього обліку ресурсів. Оскільки використання штрих-кодів є універсальним підходом до ідентифікації продукції, система може бути легко адаптована до будь-якої галузі, де потрібно швидко ідентифікувати об’єкти за кодом. Крім того, використання відкритих стандартів, API та SQLite дозволяє з мінімальними витратами розгортати

систему в будь-якому середовищі.

У перспективі продовження дослідження можливе в напрямку розширення підтримки різних форматів кодів (QR, DataMatrix, PDF417 тощо), інтеграції з хмарними базами даних, впровадження багатокористувацької авторизації, створення веб-додатка з аналогічною функціональністю, а також реалізації повноцінного мобільного додатку для роботи зі сканером у польових умовах. Крім того, перспективним є застосування штучного інтелекту для автоматичного розпізнавання товарів за зображенням або для оптимізації облікових операцій.

ПЕРЕЛІК ПОСИЛАНЬ

1. Trappey A J C, Trappey C V, Govindarajan U H, Sun J J, Chuang A C. “A review of essential standards and patent landscapes for the internet of things: A key enabler for industry 4.0.” Advanced Engineering Informatics, 2017; 33: 208–229

2. Bodnár P, Grósz T, Tóth L, Nyúl L G. Efficient visual code localization with neural networks. Pattern Analysis Applications, 2018; 21(1): 249–260.

3. Chowdhury A I, Rahman M S, Sakib N. A study of multiple barcode detection from an image in business system. International Journal of Computer Applications (0975-8887), 2019; 181(37): 30–37.

4. Zhang H, Shi G, Liu L, Zhao M, Liang Z. Detection and identification method of medical label barcode based on deep learning. Eighth International Conference on Image Processing Theory, Tools and Applications (IPTA), Xi'an, China, November 7-10, 2018. DOI: 10.1109/IPTA.2018.8608144. Accessed on [2019-07-14]

5. Beskorovainyi V. Combined method of ranking options in project decision support systems // Innovative Technologies and Scientific Solutions for Industries. 2020. No 4 (14). P. 13–20.

6. Beskorovainyi V., Petryshyn L., Shevchenko O. Specific subset effective option in technology design decisions // Applied Aspects of Information Technology. 2020. Vol. 3. No.1. P. 443–455.

7. Scandit [Електронний ресурс] – Режим доступу до ресурсу: https://www.scandit.com/.

8. Barcode Scanner [Електронний ресурс] – Режим доступу до ресурсу: https://myqrcode.com/barcode-scanner.

9. Shopify POS [Електронний ресурс] – Режим доступу до ресурсу: https://www.shopify.com/tw/pos.

10. Zebra Technologies [Електронний ресурс] – Режим доступу до ресурсу: https://www.zebra.com/de/de.html.

11. Honeywell [Електронний ресурс] – Режим доступу до ресурсу: https://www.uttc.ua/ua/vendors/honeywell/.

12. Cognex [Електронний ресурс] – Режим доступу до ресурсу: https://www.cognex.com/.

13. WebScan [Електронний ресурс] – Режим доступу до ресурсу: https://webscan.si/en.

14. QR Code Scanner by Online Barcode Reader [Електронний ресурс] – Режим доступу до ресурсу: https://online-barcode-reader.inliteresearch.com/.

15. SAP Business One [Електронний ресурс] – Режим доступу до ресурсу: https://www.sap.com/ukraine/products/erp/business-one.html.

16. Oracle NetSuite [Електронний ресурс] – Режим доступу до ресурсу: https://academy.oracle.com/ru/solutions-cloud-netsuite.html.

17. UML [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Unified_Modeling_Language.

18. PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/.

19. SQLite [Електронний ресурс] – Режим доступу до ресурсу: https://www.sqlite.org/.

20. Tkinter [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/uk/3.13/library/tkinter.html.

ДОДАТОК А

Текст програми веб-орієнтованої комп’ютерної системи з інтеграцією API для сканування штрих-кодів різних форматів

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

ВЕБ-ОРІЄНТОВАНОЇ КОМП’ЮТЕРНОЇ СИСТЕМИ З ІНТЕГРАЦІЄЮ API ДЛЯ СКАНУВАННЯ ШТРИХ-КОДІВ РІЗНИХ ФОРМАТІВ

Текст програми

804.02070743.25018-01 12 01

Листів 16

2025

АНОТАЦІЯ

Дана програма представляє собою веб-орієнтовану комп'ютерну систему, що забезпечує сканування штрих-кодів різних форматів завдяки інтеграції з відповідними API.

Програма призначена для збору та обробки даних зі штрих-кодів, що дозволяє автоматизувати процеси обліку, інвентаризації, контролю доступу або інших операцій, де необхідна швидка та точна ідентифікація об'єктів за допомогою штрих-кодів. Система підтримує різноманітні формати штрих-кодів, забезпечуючи гнучкість у застосуванні.

Програма розроблена з використанням веб-технологій, що гарантує її доступність з будь-якого пристрою, підключеного до інтернету, та забезпечує зручний користувацький інтерфейс. Вона може бути використана в різних сферах, таких як роздрібна торгівля, логістика, складський облік, бібліотечна справа та інші, де потрібно ефективне управління даними за допомогою штрих-кодів.

ЗМІСТ

Database.py

```
IMPORT RANDOM
FROM BARCODE IMPORT EAN13
FROM BARCODE.WRITER IMPORT IMAGEWRITER
FROM PIL IMPORT IMAGE
IMPORT SQLITE3

CLASS PRODUCTDATABASE:
    """КЛАС ДЛЯ КЕРУВАННЯ БАЗОЮ ДАНИХ ТОВАРІВ."""
    DEF __INIT__(SELF, DB_NAME="PRODUCTS.DB"):
        SELF.CONN = SQLITE3.CONNECT(DB_NAME)
        SELF.CURSOR = SELF.CONN.CURSOR()
        SELF._CREATE_TABLE()

    DEF _CREATE_TABLE(SELF):
        """СТВОРЮЄ ТАБЛИЦЮ ТОВАРІВ, ЯКЩО ЇЇ НЕМАЄ."""
        SELF.CURSOR.EXECUTE("""
            CREATE TABLE IF NOT EXISTS PRODUCTS (
                BARCODE TEXT PRIMARY KEY,
                NAME TEXT NOT NULL,
                PRICE REAL NOT NULL,
                QUANTITY INTEGER NOT NULL,
                IMAGE_FILENAME TEXT
            )
        """)
        SELF.CONN.COMMIT()

    DEF ADD_PRODUCT(SELF, BARCODE, NAME, PRICE, QUANTITY,
IMAGE_FILENAME):
        """ДОДАЄ НОВИЙ ТОВАР ДО БАЗИ ДАНИХ."""
```

```
            TRY:
                SELF.CURSOR.EXECUTE("INSERT INTO PRODUCTS (BARCODE, NAME, PRICE, QUANTITY, IMAGE_FILENAME) VALUES (?, ?, ?, ?, ?)", (BARCODE, NAME, PRICE, QUANTITY, IMAGE_FILENAME))
                SELF.CONN.COMMIT()
                RETURN TRUE
            EXCEPT SQLITE3.INTEGRITYERROR:
                PRINT(F"ПОМИЛКА: ШТРИХКОД '{BARCODE}' ВЖЕ ІСНУЄ В БАЗІ ДАНИХ.")
                RETURN FALSE

        DEF CLOSE(SELF):
            """ЗАКРИВАЄ З'ЄДНАННЯ З БАЗОЮ ДАНИХ."""
            SELF.CONN.CLOSE()

    DEF GENERATE_BARCODE_IMAGE(BARCODE_NUMBER, FILENAME):
        """ГЕНЕРУЄ ЗОБРАЖЕННЯ ШТРИХКОДУ ТА ЗБЕРІГАЄ ЯК JPG."""
        TRY:
            EAN = EAN13(BARCODE_NUMBER, WRITER=IMAGEWRITER())
            FULLNAME = EAN.SAVE(FILENAME.REPLACE(".JPG", "")) # БІБЛІОТЕКА САМА ДОДАЄ РОЗШИРЕННЯ
            PRINT(F"ШТРИХКОД ДЛЯ '{BARCODE_NUMBER}' ЗБЕРЕЖЕНО ЯК '{FULLNAME}.JPG'")
            RETURN TRUE
        EXCEPT EXCEPTION AS E:
            PRINT(F"ПОМИЛКА ГЕНЕРАЦІЇ ШТРИХКОДУ '{BARCODE_NUMBER}': {E}")
            RETURN FALSE

    IF __NAME__ == "__MAIN__":
```

```
BASE_PRODUCT_NAMES = [
  "ТОВАР"
]
NUM_ТОВАРІВ = 100
DB = PRODUCTDATABASE()

PRINT(F"ГЕНЕРАЦІЯ ТА ЗБЕРЕЖЕННЯ ШТРИХКОДІВ ТА ДОДАВАННЯ ДО БД (З ІМЕНАМИ ФАЙЛІВ) ({DB.CONN.TOTAL_CHANGES} ЗМІН).")

FOR I IN RANGE(1, NUM_ТОВАРІВ + 1):
  # ГЕНЕРУЄМО ВИПАДКОВИЙ ШТРИХКОД EAN-13
  BARCODE_BASE = "482" + "".JOIN(RANDOM.CHOICE("0123456789") FOR _ IN RANGE(9))
  MULTIPLIERS = [1, 3] * 6
  WEIGHTED_SUM = SUM(INT(D) * M FOR D, M IN ZIP(BARCODE_BASE, MULTIPLIERS))
  CHECKSUM = (10 - (WEIGHTED_SUM % 10)) % 10
  BARCODE = BARCODE_BASE + STR(CHECKSUM)
  PRODUCT_NAME = F"{BASE_PRODUCT_NAMES[0]} {I}"
  PRICE = ROUND(RANDOM.UNIFORM(10, 1000), 2)
  QUANTITY = RANDOM.RANDINT(1, 500)
  IMAGE_FILENAME = F"BARCODE_{BARCODE}.JPG"

  GENERATE_BARCODE_IMAGE(BARCODE, IMAGE_FILENAME)
  DB.ADD_PRODUCT(BARCODE, PRODUCT_NAME, PRICE, QUANTITY, IMAGE_FILENAME)

DB.CLOSE()
PRINT("ГЕНЕРАЦІЯ, ЗБЕРЕЖЕННЯ ШТРИХКОДІВ ТА ДОДАВАННЯ ДО
```

БД (З ІМЕНАМИ ФАЙЛІВ) ЗАВЕРШЕНО.")

Scanner.py

```
IMPORT TKINTER AS TK
FROM TKINTER IMPORT FILEDIALOG
FROM TKINTER IMPORT MESSAGEBOX
IMPORT PIL.IMAGE, PIL.IMAGETK
IMPORT TKINTER.SIMPLEDIALOG AS SIMPLEDIALOG
IMPORT SQLITE3
IMPORT RANDOM
IMPORT TKINTER.SIMPLEDIALOG AS SD
IMPORT CV2
FROM PIL IMPORT IMAGE AS PIL_IMAGE
FROM PIL IMPORT IMAGETK
IMPORT NUMPY AS NP
IMPORT WEBBROWSER



CLASS BARCODESCANNERAPP:
    DEF __INIT__(SELF, ROOT):
        SELF.ROOT = ROOT
        SELF.ROOT.TITLE("ІМІТАТОР ШТРИХ-КОД СКАНЕРА")
        SELF.DB = PRODUCTDATABASE()
        SELF.CURRENT_BARCODE = NONE
        SELF.CURRENT_PRODUCT_INFO = NONE
        IF SELF.DB.IS_EMPTY():
            SELF.POPULATE_DATABASE(100)
        SELF.CAP = NONE  # ОБ'ЄКТ ВІДЕОПОТОКУ З КАМЕРИ
        SELF.CREATE_WIDGETS()  # ВИКЛИКАЄМО CREATE_WIDGETS
ПІСЛЯ ВИЗНАЧЕННЯ ВСІХ МЕТОДІВ
```

```
    DEF MANUAL_INPUT(SELF):
        BARCODE = SIMPLEDIALOG.ASKSTRING("ВВЕДІТЬ ШТРИХ-КОД", "ВВЕДІТЬ ШТРИХ-КОД:")
        IF BARCODE:
            SELF.CHECK_AND_DISPLAY_PRODUCT(BARCODE)

    DEF LOAD_AND_ASK_BARCODE(SELF):
        FILE_PATH = FILEDIALOG.ASKOPENFILENAME(FILETYPES=[("IMAGE FILES", "*.PNG;*.JPG;*.JPEG;*.BMP")])
        IF FILE_PATH:
            # ЗАВАНТАЖЕННЯ ЗОБРАЖЕННЯ ЗА ДОПОМОГОЮ OPENCV
            IMAGE = CV2.IMREAD(FILE_PATH)
            DETECTOR = CV2.QRCODEDETECTOR()
            DATA, BBOX, _ = DETECTOR.DETECTANDDECODE(IMAGE)

            SELF.DISPLAY_IMAGE(FILE_PATH)  # ПОКАЗАТИ ЗОБРАЖЕННЯ В ІНТЕРФЕЙСІ

            IF DATA:
                # ЯКЩО КОД РОЗПІЗНАНО — ПЕРЕВІРИТИ ТА ПОКАЗАТИ ІНФОРМАЦІЮ ПРО ТОВАР
                SELF.CHECK_AND_DISPLAY_PRODUCT(DATA)
            ELSE:
                SELF.RESULT_DISPLAY.CONFIG(TEXT="QR/ШТРИХ-КОД НЕ РОЗПІЗНАНО. ВВЕДІТЬ ВРУЧНУ.", FG="RED")

    DEF CHECK_AND_DISPLAY_PRODUCT(SELF, BARCODE):
        PRODUCT = SELF.DB.GET_PRODUCT(BARCODE)
        IF PRODUCT:
```

```
            SELF.CURRENT_BARCODE = BARCODE
            SELF.CURRENT_PRODUCT_INFO = {"NAME": PRODUCT[0], "PRICE": PRODUCT[1], "QUANTITY": PRODUCT[2]}
            NAME, PRICE, QUANTITY = PRODUCT
            RESULT_TEXT = F"ШТРИХ-КОД: {BARCODE}\NТОВАР: {NAME}\NЦІНА: {PRICE} ГРН\NЗАЛИШОК: {QUANTITY} ШТ"
            SELF.RESULT_DISPLAY.CONFIG(TEXT=RESULT_TEXT, FG="GREEN", JUSTIFY="LEFT")
            SELF.ADD_TO_CART_BUTTON.CONFIG(STATE=TK.NORMAL)
            SELF.COPY_TO_CLIPBOARD(F"{BARCODE} - {NAME} - {PRICE} ГРН - {QUANTITY} ШТ")
        ELSE:
            SELF.CURRENT_BARCODE = NONE
            SELF.CURRENT_PRODUCT_INFO = NONE
            SELF.RESULT_DISPLAY.CONFIG(TEXT=F"ШТРИХ-КОД: {BARCODE}\NТОВАР НЕ ЗНАЙДЕНО", FG="RED", JUSTIFY="LEFT")
            SELF.ADD_TO_CART_BUTTON.CONFIG(STATE=TK.DISABLED)
            SELF.COPY_TO_CLIPBOARD(BARCODE)

    DEF ADD_TO_CART(SELF):
        IF SELF.CURRENT_BARCODE AND SELF.CURRENT_PRODUCT_INFO:
            QUANTITY_TO_ADD = SD.ASKINTEGER("ДОДАТИ В КОШИК", F"ВВЕДІТЬ КІЛЬКІСТЬ ДЛЯ '{SELF.CURRENT_PRODUCT_INFO['NAME']}' (ДОСТУПНО {SELF.CURRENT_PRODUCT_INFO['QUANTITY']}):", MINVALUE=1, MAXVALUE=SELF.CURRENT_PRODUCT_INFO['QUANTITY'])
            IF QUANTITY_TO_ADD IS NOT NONE:
                IF SELF.DB.DECREASE_QUANTITY(SELF.CURRENT_BARCODE, QUANTITY_TO_ADD):
                    MESSAGEBOX.SHOWINFO("КОШИК", F"{QUANTITY_TO_ADD} ОД. '{SELF.CURRENT_PRODUCT_INFO['NAME']}'
```

```
ДОДАНО ДО КОШИКА.")
                    # ОНОВЛЮЄМО ВІДОБРАЖЕННЯ ЗАЛИШКУ
                    PRODUCT                                                    =
SELF.DB.GET_PRODUCT(SELF.CURRENT_BARCODE)
                    IF PRODUCT:
                      SELF.CURRENT_PRODUCT_INFO['QUANTITY']                    =
PRODUCT[2]
                      NAME, PRICE, QUANTITY = PRODUCT
                      RESULT_TEXT          =          F"ШТРИХ-КОД:
{SELF.CURRENT_BARCODE}\NТОВАР:          {NAME}\NЦІНА:          {PRICE}
ГРН\NЗАЛИШОК: {QUANTITY} ШТ"
                      SELF.RESULT_DISPLAY.CONFIG(TEXT=RESULT_TEXT,
FG="GREEN", JUSTIFY="LEFT")
                      IF QUANTITY == 0:

SELF.ADD_TO_CART_BUTTON.CONFIG(STATE=TK.DISABLED)
                    ELSE:
                      SELF.RESULT_DISPLAY.CONFIG(TEXT="ПОМИЛКА: ТОВАР
НЕ ЗНАЙДЕНО В БД.", FG="RED")

SELF.ADD_TO_CART_BUTTON.CONFIG(STATE=TK.DISABLED)
                ELSE:
                  MESSAGEBOX.SHOWERROR("ПОМИЛКА",    "НЕДОСТАТНЬО
ТОВАРУ НА СКЛАДІ.")
            ELSE:
              MESSAGEBOX.SHOWINFO("ІНФОРМАЦІЯ",          "СПОЧАТКУ
СКАНУЙТЕ АБО ВВЕДІТЬ ШТРИХ-КОД ТОВАРУ.")

        DEF DISPLAY_IMAGE(SELF, IMAGE_PATH):
          TRY:
```

```
            IMAGE = PIL.IMAGE.OPEN(IMAGE_PATH)
            IMAGE.THUMBNAIL((400, 400))
            IMAGE_TK = PIL.IMAGETK.PHOTOIMAGE(IMAGE)
            SELF.LABEL.CONFIG(IMAGE=IMAGE_TK)
            SELF.LABEL.IMAGE = IMAGE_TK
        EXCEPT EXCEPTION AS E:
            MESSAGEBOX.SHOWERROR("ПОМИЛКА", F"НЕ ВДАЛОСЯ ВІДКРИТИ ЗОБРАЖЕННЯ: {E}")

    DEF COPY_TO_CLIPBOARD(SELF, TEXT):
        SELF.ROOT.CLIPBOARD_CLEAR()
        SELF.ROOT.CLIPBOARD_APPEND(TEXT)
        SELF.ROOT.UPDATE()

    DEF POPULATE_DATABASE(SELF, NUM_PRODUCTS=100):
        IF NOT SELF.DB.IS_EMPTY():
            RETURN
        BASE_NAMES = ["ЯБЛУКО", "БАНАН", "АПЕЛЬСИН", "МОЛОКО", "ХЛІБ", "СИР", "КОВБАСА", "МАКАРОНИ", "РИС", "ЦУКОР"]
        SORTS = ["ВИЩИЙ", "ПЕРШИЙ", "ЕКСТРА", "СЕЛЯНСЬКИЙ", "ФЕРМЕРСЬКИЙ"]
        FOR I IN RANGE(NUM_PRODUCTS):
            BARCODE = F"999{RANDOM.RANDINT(10000000000, 99999999999)}"
            NAME = F"{RANDOM.CHOICE(BASE_NAMES)} ({RANDOM.CHOICE(SORTS)})"
            PRICE = ROUND(RANDOM.UNIFORM(10, 500), 2)
            QUANTITY = RANDOM.RANDINT(10, 200)
            SELF.DB.ADD_PRODUCT(BARCODE, NAME, PRICE, QUANTITY)
        MESSAGEBOX.SHOWINFO("ІНФОРМАЦІЯ", F"БАЗУ ДАНИХ
```

```
ЗАПОВНЕНО {NUM_PRODUCTS} ВИПАДКОВИМИ ТОВАРАМИ.")

    DEF OPEN_CAMERA(SELF):
        SELF.CAP = CV2.VIDEOCAPTURE(0)
        IF NOT SELF.CAP.ISOPENED():
            MESSAGEBOX.SHOWERROR("ПОМИЛКА", "НЕ ВДАЛОСЯ
ВІДКРИТИ КАМЕРУ.")
            RETURN
        SELF.UPDATE_FRAME()

    DEF UPDATE_FRAME(SELF):
        IF NOT SELF.CAP:
            RETURN
        RET, FRAME = SELF.CAP.READ()
        IF RET:
            GRAY = CV2.CVTCOLOR(FRAME, CV2.COLOR_BGR2GRAY)
            FRAME_PIL = PIL_IMAGE.FROMARRAY(CV2.CVTCOLOR(FRAME,
CV2.COLOR_BGR2RGB))
            FRAME_TK = IMAGETK.PHOTOIMAGE(FRAME_PIL)
            SELF.CAMERA_FRAME.CONFIG(IMAGE=FRAME_TK)
            SELF.CAMERA_FRAME.IMAGE = FRAME_TK
        SELF.ROOT.AFTER(10, SELF.UPDATE_FRAME)

    DEF CLOSE_CAMERA(SELF):
        IF SELF.CAP AND SELF.CAP.ISOPENED():
            SELF.CAP.RELEASE()
            SELF.CAP = NONE
            SELF.CAMERA_FRAME.CONFIG(IMAGE='')
            SELF.CAMERA_FRAME.IMAGE = NONE
```

```
    DEF ON_CLOSING(SELF):
        SELF.CLOSE_CAMERA()
        SELF.ROOT.DESTROY()

    DEF OPEN_WEB_SCANNER(SELF):
        """ВІДКРИВАЄ ВЕБ-СТОРІНКУ СКАНЕРА ШТРИХ-КОДІВ У БРАУЗЕРІ."""

WEBBROWSER.OPEN_NEW("HTTPS://GEORAPBOX.GITHUB.IO/BARCODE-SCANNER/")

    DEF CREATE_WIDGETS(SELF):
        SELF.LABEL = TK.LABEL(SELF.ROOT, TEXT="IMITATOP ШТРИХ-КОД СКАНЕРА", FONT=("ARIAL", 18))
        SELF.LABEL.PACK(PADY=10)

        SELF.INPUT_BUTTON = TK.BUTTON(SELF.ROOT, TEXT="ВВЕСТИ ШТРИХ-КОД ВРУЧНУ", COMMAND=SELF.MANUAL_INPUT, WIDTH=30)
        SELF.INPUT_BUTTON.PACK(PADY=10)

        SELF.FILE_BUTTON = TK.BUTTON(SELF.ROOT, TEXT="ЗАВАНТАЖИТИ ЗОБРАЖЕННЯ", COMMAND=SELF.LOAD_AND_ASK_BARCODE, WIDTH=40)
        SELF.FILE_BUTTON.PACK(PADY=10)

        SELF.CAMERA_BUTTON = TK.BUTTON(SELF.ROOT, TEXT="ВІДКРИТИ КАМЕРУ", COMMAND=SELF.OPEN_CAMERA, WIDTH=30)
        SELF.CAMERA_BUTTON.PACK(PADY=10)

        SELF.WEB_SCANNER_BUTTON = TK.BUTTON(SELF.ROOT,
```

```
TEXT="ВЕБ СКАНЕР", COMMAND=SELF.OPEN_WEB_SCANNER, WIDTH=30)
        SELF.WEB_SCANNER_BUTTON.PACK(PADY=10)

        SELF.RESULT_LABEL = TK.LABEL(SELF.ROOT,
TEXT="РЕЗУЛЬТАТ:", FONT=("ARIAL", 12))
        SELF.RESULT_LABEL.PACK(PADY=10)

        SELF.RESULT_DISPLAY = TK.LABEL(SELF.ROOT, TEXT="",
FONT=("ARIAL", 14, "BOLD"), FG="GREEN", JUSTIFY="LEFT")
        SELF.RESULT_DISPLAY.PACK(PADY=5)

        SELF.ADD_TO_CART_BUTTON = TK.BUTTON(SELF.ROOT,
TEXT="ДОДАТИ В КОШИК", COMMAND=SELF.ADD_TO_CART,
STATE=TK.DISABLED, WIDTH=30)
        SELF.ADD_TO_CART_BUTTON.PACK(PADY=10)

        SELF.QUIT_BUTTON = TK.BUTTON(SELF.ROOT, TEXT="ВИЙТИ",
COMMAND=SELF.ROOT.QUIT, WIDTH=30)
        SELF.QUIT_BUTTON.PACK(PADY=20)

        SELF.CAMERA_FRAME = TK.LABEL(SELF.ROOT)
        SELF.CAMERA_FRAME.PACK()

    CLASS PRODUCTDATABASE:
        DEF __INIT__(SELF, DB_NAME="PRODUCTS.DB"):
            SELF.CONN = SQLITE3.CONNECT(DB_NAME)
            SELF.CURSOR = SELF.CONN.CURSOR()
            SELF._CREATE_TABLE()

        DEF _CREATE_TABLE(SELF):
```

```
        SELF.CURSOR.EXECUTE("""
            CREATE TABLE IF NOT EXISTS PRODUCTS (
                BARCODE TEXT PRIMARY KEY,
                NAME TEXT NOT NULL,
                PRICE REAL NOT NULL,
                QUANTITY INTEGER NOT NULL
            )
        """)
        SELF.CONN.COMMIT()

    DEF ADD_PRODUCT(SELF, BARCODE, NAME, PRICE, QUANTITY):
        TRY:
            SELF.CURSOR.EXECUTE("INSERT INTO PRODUCTS (BARCODE, NAME, PRICE, QUANTITY) VALUES (?, ?, ?, ?)", (BARCODE, NAME, PRICE, QUANTITY))
            SELF.CONN.COMMIT()
            RETURN TRUE
        EXCEPT SQLITE3.INTEGRITYERROR:
            PRINT(F"ПОМИЛКА: ШТРИХКОД '{BARCODE}' ВЖЕ ІСНУЄ В БАЗІ ДАНИХ.")
            RETURN FALSE

    DEF GET_PRODUCT(SELF, BARCODE):
        SELF.CURSOR.EXECUTE("SELECT NAME, PRICE, QUANTITY FROM PRODUCTS WHERE BARCODE=?", (BARCODE,))
        RESULT = SELF.CURSOR.FETCHONE()
        RETURN RESULT

    DEF DECREASE_QUANTITY(SELF, BARCODE, QUANTITY):
        SELF.CURSOR.EXECUTE("UPDATE PRODUCTS SET QUANTITY =
```

```
QUANTITY - ? WHERE BARCODE = ? AND QUANTITY >= ?", (QUANTITY, BARCODE, QUANTITY))
            SELF.CONN.COMMIT()
            RETURN SELF.CURSOR.ROWCOUNT > 0

        DEF IS_EMPTY(SELF):
            SELF.CURSOR.EXECUTE("SELECT COUNT(*) FROM PRODUCTS")
            COUNT = SELF.CURSOR.FETCHONE()[0]
            RETURN COUNT == 0

        DEF CLOSE(SELF):
            SELF.CONN.CLOSE()

    IF __NAME__ == "__MAIN__":
        ROOT = TK.TK()
        APP = BARCODESCANNERAPP(ROOT)
        ROOT.PROTOCOL("WM_DELETE_WINDOW", APP.ON_CLOSING)
        ROOT.MAINLOOP()
```