Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра комп'ютерних інтелектуальних технологій та систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Інтелектуальна система аналізу якості продуктів харчування за зображенням

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-10
Даценко Денис Сергійович
(власне ім'я, прізвище)

Спеціальність
123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп'ютерна інженерія
(повна назва освітньої програми)

Керівник: ас. каф. Андрій ТАТАРНИКОВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Олег РУДЕНКО
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра комп'ютерних інтелектуальних технологій та систем

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 2025 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Даценку Денису Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інтелектуальна система аналізу якості продуктів харчування за зображенням

затверджена наказом по університету від "21" травня 2025 р. № 399 Ст.

2. Термін подання здобувачем роботи до екзаменаційної комісії 11 червня 2025 р.

3. Вхідні дані до роботи

1. Документація мови програмування Java та фреймворку Spring Boot.
2. Документація Java API TensorFlow та OpenCV.
3. Інтегроване середовище розробки IntelliJ IDEA.
4. Документація HTML, CSS, Tailwind та JavaScript.
5. Середовище розробки Visual Studio Code.
6. Браузерні інструменти для тестування веб-застосунку.

4. Перелік питань, що потрібно опрацювати у роботі

1. Аналіз предметної області.
2. Аналіз використовуваних технологій.
3. Програмна реалізація клієнтської та серверної частин, разом з інтелектуальним модулем.
4. Інструкція користувача.
5. Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 16 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	26.05.2025-28.05.2025	Виконано
2	Вибір технологій розробки	29.05.2025-31.05.2025	Виконано
3	Розробка підходу для реалізації завдання	01.06.2025-02.06.2025	Виконано
4	Реалізація серверної частини	03.06.2025-06.06.2025	Виконано
5	Розробка клієнтського інтерфейсу	07.06.2025-09.06.2025	Виконано
6	Інтеграція моделі машинного навчання та її	10.06.2025-11.06.2025	Виконано
	налаштування		
7	Проведення тестування функціональності	12.06.2025-13.06.2025	Виконано
	системи		
8	Оформлення матеріалів атестаційної роботи	13.06.2025-16.06.2025	Виконано
9	Подання атестаційної роботи керівникові	17.06.2025-18.06.2025	Виконано
10	Подання роботи на рецензування	19.06.2025-20.06.2025	Виконано

Дата видачі завдання "26" травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. каф. Андрій ТАТАРНИКОВ
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 67 с., 22 рис., 21 ліст., 2 табл., 2 дод., 32 джерела.

ІНТЕЛЕКТУАЛЬНА СИСТЕМА, АНАЛІЗ ЯКОСТІ, ПРОДУКТИ ХАРЧУВАННЯ, КОМП'ЮТЕРНИЙ ЗІР, НЕЙРОННІ МЕРЕЖІ, ВЕБ-ЗАСТОСУНОК, ОЦІНКА ЗОБРАЖЕНЬ.

Метою атестаційної роботи є розробка інтелектуальної системи для аналізу якості продуктів харчування на основі зображень. Система використовує методи комп'ютерного зору та глибокого навчання для автоматичного розпізнавання та оцінки стану продуктів.

В результаті проведених досліджень були визначені основні проблеми та недоліки існуючих рішень у сфері автоматизованої оцінки якості харчових продуктів, що стало основою для формування вимог до функціоналу розроблюваної системи.

Розроблений застосунок складається із серверної та клієнтської частини. Серверна частина реалізована з використанням технології Spring Boot, TensorFlow Java для виконання процесів обробки зображень та OpenCV для попередньої обробки даних. Для зберігання інформації про результати роботи застосунку використовується база даних PostgreSQL, а взаємодія з полями в БД здійснюється за допомогою Hibernate. Клієнтська частина є веб-застосунком, створеним з використанням класичних підходів веб-розробки та дозволяє користувачам завантажувати зображення продуктів та переглядати результати аналізу якості.

Запропоноване рішення спрямоване на підвищення ефективності та об'єктивності оцінки стану харчових продуктів, що може бути корисним як для споживачів, так і для підприємств харчової галузі.

ABSTRACT

Bachelor’s thesis: 67 pages, 22 figures, 21 listings, 2 tables, 2 appendices, 32 sources.

INTELLIGENT SYSTEM, QUALITY ANALYSIS, FOOD PRODUCTS, COMPUTER VISION, NEURAL NETWORKS, WEB APPLICATION, IMAGE EVALUATION.

The aim of the certification work is to develop an intelligent system for analysing food quality based on images. The system uses computer vision and deep learning methods to automatically recognise and assess the condition of products.

As a result of the research, the main problems and shortcomings of existing solutions in the field of automated food quality assessment were identified, which became the basis for the formation of requirements for the functionality of the system under development.

The developed application consists of a server and a client part. The server part is implemented using Spring Boot technology, TensorFlow Java for image processing and OpenCV for data pre-processing. The PostgreSQL database is used to store information about the application results, and interaction with the fields in the database is carried out using Hibernate. The client part is a web application created using classical web development approaches and allows users to upload product images and view the results of quality analysis.

The proposed solution is aimed at improving the efficiency and objectivity of food quality assessment, which can be useful for both consumers and food companies.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ISO – міжнародна організація зі стандартизації (англ., International Organization for Standardization)

CV – комп'ютерний зір (англ., Computer Vision)

ML – машинне навчання (англ., Machine Learning)

DL – глибоке навчання (англ., Deep Learning)

ANN – штучна нейронна мережа (англ., Artificial Neural Network)

CNN – згорткова нейронна мережа (англ., Convolutional Neural Network)

YOLO – сімейство моделей машинного навчання «Ти Дивишся Лише Раз» (англ., You Only Look Once)

API – інтерфейс програмування застосунків (англ., Application Programming Interface)

HTTP – протокол передачі гіпертексту (англ. HyperText Transfer Protocol)

UID – дизайн інтерфейсу користувача (англ., User Interface Design)

URL – уніфікований локатор ресурсів (англ., Uniform Resource Locator)

DOM – об'єктна модель документа (англ., Document Object Model)

HTML – мова розмітки гіпертексту (англ., HyperText Markup Language)

CSS – каскадні таблиці стилів (англ., Cascading Style Sheets)

REST – передача репрезентативного стану (англ., Representational State Transfer)

ORM – об'єктно-реляційна проєкція (англ., Object-Relational Mapping)

SQL – мова структурованих запитів (англ., Structured query language)

OpenCV – бібліотека комп'ютерного зору з відкритим кодом (англ., Open Source Computer Vision Library)

JVM – віртуальна машина Java (англ., Java Virtual Machine)

ВСТУП

Якість харчових продуктів є одним із ключових факторів, що впливають на здоров’я людей та їхню довіру до виробників. У сучасному світі, де обсяги виробництва та постачання продуктів харчування постійно зростають, важливо забезпечити ефективні методи контролю якості, що дозволяють швидко та точно оцінювати стан продукції. Традиційні методи перевірки якості часто потребують значних ресурсів, спеціального обладнання та часу, що ускладнює їх широке застосування.

Сучасні технології комп’ютерного зору надають доступ до нових можливостей для автоматизації оцінки якості харчових продуктів. Використання ШІ надає набір інструментів для провдення аналізу та обробки зображення, визначати стан продуктів на них, виявляти дефекти та прогнозувати термін придатності. Такі підходи можуть значно спростити процес контролю якості на виробництвах.

Для вирішення цієї проблеми пропонується розробити інтелектуальну систему аналізу якості продуктів харчування за зображенням. Система базується на методах машинного навчання та комп’ютерного зору та включає такі основні функції:

- обробка та аналіз зображень продуктів харчування;
- використання глибоких нейронних мереж для оцінки якості продукції;
- інтеграція веб-застосунку для взаємодії з користувачами;
- надання результатів аналізу у зручному форматі.

Запропонована система спрямована на підвищення ефективності перевірки якості харчових продуктів, що може бути корисним як для підприємств харчової промисловості, так і для кінцевих споживачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проблеми забезпечення якості харчових продуктів

За останні десятиліття харчова промисловість переживає стрімке зростання та диверсифікацію, зумовлені збільшенням світового попиту, підвищенням стандартів безпечності харчових продуктів та підвищенням рівня обізнаності споживачів. Забезпечення якості та безпечності харчових продуктів по всьому ланцюгу постачання – від виробництва та пакування до зберігання та дистрибуції – стало критично важливою вимогою. Споживачі бажають не лише смачної та поживної їжі, але й свіжої, візуально привабливої, без псування чи забруднення. Як наслідок, оцінка якості харчових продуктів стала головним питанням у харчовій індустрії [1].

Традиційно контроль якості харчових продуктів ґрунтувався на перевірці, яку здійснювали люди. Хоч такий підхід і може бути до певної міри ефективним, він страждає від низки суттєвих обмежень. Людська оцінка є суб'єктивною, схильною до втоми і відволікаючих факторів. Крім того, у великих промислових масштабах така перевірка дуже повільна та потребувати багато ресурсів, щоб відповідати вимогам пропускної здатності.

1.2 Визначення якості в харчовій промисловості

Забезпечення якості харчових продуктів є життєво важливим аспектом сучасного ланцюга постачання харчових продуктів, що впливає не лише на задоволення потреб споживачів, але й на здоров'я населення. Оцінка якості харчових продуктів передбачає визначення того, чи відповідає продукт встановленим стандартам свіжості, безпечності та товарного вигляду.

Якість харчових продуктів представляє собою поєднання сенсорних, поживних, безпечних та естетичних характеристик, які визначають

прийнятність. Для свіжих продуктів, таких як фрукти та овочі, ключовими показниками якості є свіжість, стиглість та відсутність зіпсованості.

Свіжість – наскільки недавно продукт був зібраний або перероблений, і часто асоціюється з візуальними ознаками, такими як яскравий колір і пружність. Стиглість відображає оптимальну для споживання стадію, коли продукт досягає свого піку за смаком, текстурою та поживною цінністю. Псування, в свою чергу, вказує на деградацію через ферментативну активність, мікробне забруднення або хімічні зміни, що робить продукт небезпечним для споживання. Зміна кольору, наявність плісняви, усадка або деформація є типовими симптомами псування або пошкодження. Ці характеристики оцінюються візуально, що робить їх придатними для аналізу на основі зображень [2].

1.3 Існуючі недоліки в класичних методах оцінки та контролю якості

Забезпечення необхідного рівня контролю за якістю продуктів зазвичай виконується кваліфікованим персоналом, який виконує візуальну оцінку. Такий підхід може працювати на невеликих підприємствах, але у промислових умовах створюється низка проблем через суб'єктивність та схильність до людських помилок та неуважності. Візуальна перевірка людиною займає багато часу і не може йти в ногу з високошвидкісними виробничими лініями. Безперервний контроль якості вимагає великого обсягу людських ресурсів, особливо на великих підприємствах, чого не неможливо досягти таким підходом до контролю [3].

Постачання неякісних або небезпечних харчових продуктів може призвести до незадоволення споживачів. Недотримання вимог щодо безпечності та якості харчових продуктів може призвести до юридичних санкцій, відкликання або заборони на розповсюдження продукції, що вплине на економічну складову. Міжнародні стандарти, такі як ISO 22000, підкреслюють необхідність надійних механізмів контролю якості [4].

Отже, сучасні виробники харчових продуктів перебувають під дедалі більшим тиском і змушені впроваджувати технології, які забезпечують точний контроль оцінки якості їхньої продукції.

1.4 Сучасні підходи до візуальної оцінки якості продуктів

Оскільки харчову промисловість все більше охоплює автоматизація, комп'ютерний зір став потужним інструментом для оцінки якості продукції. Computer Vision – це підгалузь, яка надає можливості для вилучення, обробки та інтерпретації візуальної інформації з цифрових зображень за допомогою методів ШІ. В аналізі харчових продуктів дана технологія імітує людську зорову систему для оцінки критичних параметрів якості, таких як колір, текстура, форма і поверхневі дефекти [5].

У контексті оцінки якості системи комп'ютерного зору, як правило, працюють у кілька етапів. Спочатку створюються проміжні рисунки продукту з попередніми налаштуваннями націленими на високу якість.

Рисунок 1.1 – Приклад набору даних із зображеннями фруктів

Ці зображення проходять етапи попередньої обробки, такі як зменшення шуму, нормалізація кольору і підвищення контрастності, щоб забезпечити узгодженість і виділити відповідні характеристики. Після цього

за допомогою методів обробки зображень виділяються ключові візуальні атрибути які були вилучені із завантажених зображень.

Такі системи покладаються на визначені розробником правила для виявлення та класифікації ознак. Щоб подолати ці обмеження, сучасні підходи все частіше звертаються до моделей на основі даних, які створюються за допомогою ML та DL та пропонують більшу гнучкість і точність. Машинне навчання підвищує адаптивність систем візуального контролю, дозволяючи їм навчатися на основі даних [6].

Серед архітектур глибокого навчання згорткові нейронні мережі доказали свою високу ефективність в завданнях, де необхідно виконувати операції класифікації зображень, розпізнавання об'єктів і виявлення дефектів. Ці моделі автоматично вивчають багаторівневі представлення даних зображень, від низькорівневих країв і текстур до високорівневих семантичних ознак, без зовнішнього втручання. Попередньо навчені мережі, такі як ResNet, EfficientNet і MobileNet, можуть бути точно налаштовані на відносно невеликих наборах даних зображень харчових продуктів, що скорочує час розробки і покращує продуктивність узагальнення [7].

Інтегруючи комп'ютерний зір з методами ML і DL, сучасні системи інспекції харчових продуктів можуть досягти більшої точності, працювати в режимі реального часу і покращити масштабованість – трансформуючи процеси забезпечення якості в ланцюгах постачання харчових продуктів.

1.5 Існуючі дослідження та рішення

Протягом останнього десятиліття численні ініціативи та розробки були зосереджені на застосуванні комп'ютерного зору та методів DL до проблеми оцінки якості харчових продуктів. Ці рішення продемонстрували великий потенціал у підвищенні точності, швидкості та масштабованості візуальних перевірок, завдяки чому отримали визнання та стали стандартом розробки подібних систем.

1.5.1 Наукові дослідження

У науковому середовищі активно розробляються та тестуються прототипи, що застосовують CNN. Наукові дослідження з використанням ResNet та MobileNet показують, що за зображенням можна з високою точністю класифікувати ступінь свіжості яблук, бананів, томатів тощо. Одним з яскравих прикладів є застосування ЗНМ для класифікації фруктів. Наприклад, у своєму дослідженні Matyam Rahnemoonfar та Clay Sheppard у 2017 році продемонстрували систему на основі ШНМ, яка успішно виявила помідори і оцінила рівень їх стиглості в природних умовах, досягнувши високої точності навіть в умовах оклюзії і різного освітлення [8].

Використовувані передові засоби, такі як YOLO та його аналоги, також використовувалися для отримання інформації про дефекти на овочах. Швидкість і точність YOLO роблять її придатною для вбудованих застосунків, де необхідне швидке прийняття рішень. Паралельно з цим, легкі архітектури, такі як MobileNet і EfficientNet, дозволили розгортати моделі класифікації харчових продуктів на мобільних і периферійних пристроях, полегшуючи для споживачів інтеграцію систем в свої виробництва.

У таблиці 1.1 представлено порівняльний огляд найпоширеніших підходів, що використовуються для виконання візуальної оцінки якості харчових продуктів.

Таблиця 1.1 – Порівняння сучасних методів для візуальної оцінки якості

Характеристики	Методи		
	CV	ML	DL
Точність	Середня	Висока	Дуже висока
Швидкість	Висока	Середня	Висока
Масштабованість	Обмежена	Помірна	Висока
Вартість	Низька	Помірна	Висока
Складність впровадження	Низька	Помірна	Висока

Роботи, опубліковані на тематичних ресурсах, демонструють використання моделей машинного навчання у поєднанні з гіперспектральними або RGB-зображеннями. Такі підходи дозволяють досягати точності понад 90% при класифікації якості, але часто вимагають специфічних умов зйомки, потужного обчислювального обладнання або попередньої обробки зображень [9].

1.6 Комерційні рішення

У зв’язку з нагальною необхідністю підвищити точність, швидкодію та об’єктивність оцінювання продуктів харчування, на ринку з’явилося багато готових до використання комерційних рішень.

1.6.1 Clarifruit

Clarifruit – автоматизована платформа на основі штучного інтелекту для контролю якості свіжої продукції без втручання людини.

Рисунок 1.2 – Інтерфейс Clarifruit

Рішення Clarifruit інтегрує комп'ютерний зір, машинне навчання та мобільні технології для оцінки зовнішніх атрибутів продукту з високим ступенем точності та повторюваності.

Платформа дозволяє користувачам використовувати смартфони, щоб отримувати зображення фруктів в звичайних умовах за допомогою інтуїтивно зрозумілого інтерфейсу, що наведений на рисунку 1.2.

Потім ці зображення обробляються хмарним механізмом Clarifruit, який оцінює ключові параметри якості. Аналіз базується на власних моделях, первинне навчання яких було проведено на попередньо підготованих даних. За лічені секунди користувачі отримують звіт про якість відповідно до попередньо налаштованих стандартів оцінювання, що дозволяє приймати рішення в автоматичному режимі [10].

Попри свою інноваційність, дане рішення має ряд обмежень. Зокрема, система демонструє низьку гнучкість та можливість інтеграції нових або локальних плодів без потреби в донавчанні моделей. Крім того, закритість програмної архітектури не дозволяє користувачам здійснювати глибоку адаптацію або інтеграцію з іншими сервісами. Інтерфейс не забезпечує достатніх можливостей для формування розгорнутої аналітики або відображення ключових показників у реальному часі.

1.6.2 Agroscout

Agroscout – застосунок, що спеціалізується на землеробстві та дистанційному моніторингу посівів за допомогою штучного інтелекту. На відміну від систем, орієнтованих виключно на контроль якості після збору врожаю, Agroscout забезпечує наскрізний моніторинг протягом усього життєвого циклу культури, пропонуючи цінну інформацію про здоров'я рослин, яка в свою чергу впливає на зовнішній вигляд товару. Платформа поєднує в собі зображення з дронів і мобільних пристроїв з хмарними алгоритмами машинного навчання, що дає можливість раннього виявлення, що має позитивні результати в рамках підвищення якості продуктів [11].

В основі технології Agroscout лежить здатність аналізувати повітряні та близькі зображення, зібрані з дронів або смартфонів. Ці зображення

завантажуються на хмарну платформу компанії, де алгоритми комп'ютерного зору виявляють аномалії. Містить зрозумілий веб-інтерфейс та мобільний застосунок для моніторингу, що наведений на рисунку 1.3.

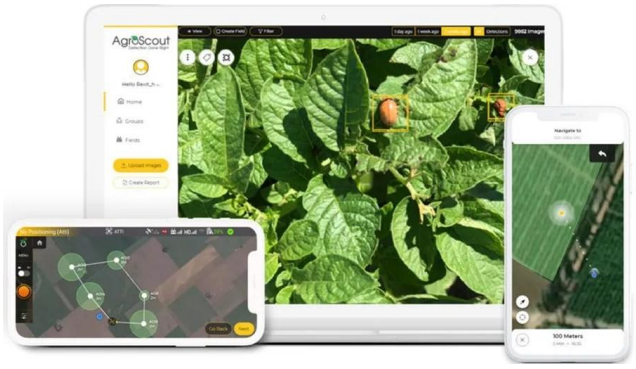

Рисунок 1.3 – Користувацький інтерфейс Agroscout

Основним недоліком рішення є його орієнтація на агрономічний моніторинг, а не на післязбиральний аналіз якості продукції. Система не передбачає детальну оцінку свіжості чи виявлення дефектів овочів і фруктів. Додатково, для повноцінного функціонування необхідне спеціалізоване обладнання. Аналіз результатів обмежується польовими умовами, що не забезпечує повного контролю якості на етапі логістики чи зберігання.

1.6.3 Tomra Food

Tomra Food – це провідна міжнародна компанія, яка спеціалізується на розробці та впровадженні високотехнологічних рішень для автоматизованого сортування, класифікації та контролю якості харчових продуктів.

Tomra Food пропонує сортувальні машини, які працюють у режимі реального часу, з можливістю інтеграції у виробничі лінії. Системи компанії здатні ідентифікувати навіть незначні дефекти або забруднення – наприклад, внутрішні ушкодження фруктів чи мікроскопічні плями на шкірці. Важливо, що ці технології не тільки підвищують ефективність і швидкість сортування,

а й суттєво покращують якість кінцевої продукції, зменшуючи втрати [12].

Крім того, завдяки впровадженню аналітики даних, користувачі можуть відстежувати статистику і приймати рішення на основі точних показників якості в реальному часі, як от на рисунку 1.4.

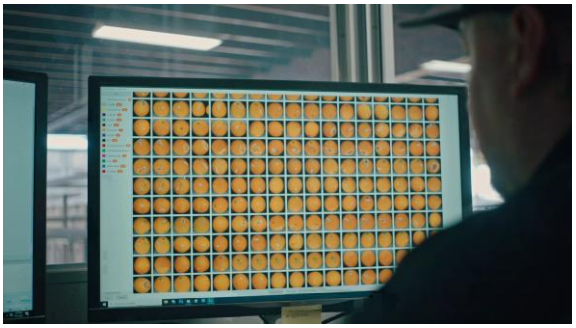

Рисунок 1.4 – Робота з інтерфейсом Tomra Invision

Незважаючи на високу точність роботи, основними недоліками є висока вартість впровадження та обслуговування, що знижує доступність для малих і середніх підприємств. Платформа орієнтована переважно на механічне сортування, без надання гнучкого інструментарію для поглибленого аналізу якості чи формування інтерактивних звітів.

1.7 Результати аналізу існуючих рішень

Серед розглянутих аналогів, що можуть використовуватися для контролю якості, можна визначити унікальні концепції функціоналу:

– Clarifruit – перевірка якості свіжої продукції в режимі реального часу за допомогою смартфонів;

– Agroscout – моніторинг посівів на всіх стадіях розвитку агрокультури за допомогою знімків з дронів і мобільних зображень які вконютсья безпосередньо на полях у поєднанні зі штучним інтелектом;

– Tomra Food – сортування та класифікації на основі сенсорів промислового масштабу.

Частина зазначених програм не має методів для відображення метрик ефективності роботи системи в реальному часі, що ускладнює оцінку їхньої надійності при різних умовах застосування та обмежує можливість перенавчання або пристосування нейронних мереж до специфічних вимог виробництва. Результати аналізу наведені в таблиці 1.2.

Таблиця 1.2 – Порівняння характеристик програмних рішень

Характеристика	Програмні рішення		
	Clarifruit	Agroscout	Tomra
Основне призначення	Оцінка якості продукції	Моніторинг стану до збору	Сортування та оцінка якості
Технології	Deep Learning	Deep Learning	Deep Learning
Розгортання	Пристрої з камерою	Пристрої з камерою	Промислове обладнання
Швидкість	В реальному часі	З періодичністю	В реальному часі
Масштабованість	Низька	Висока	Дуже висока
Вартість	Низька	Помірна	Висока

1.8 Постановка задачі

Проєкт представляє собою веб-застосунок для аналізу якості продуктів харчування за зображеннями, зосереджений на оцінці свіжості фруктів та овочів. Основна мета – створення інтелектуальної системи, яка за допомогою нейронних мереж зможе виявляти дефекти, класифікувати ступінь придатності продуктів до споживання і автоматично відображати результати аналізу через зручний веб-інтерфейс.

На основі аналізу існуючих рішень у сфері контролю якості продуктів було визначено потребу у розробці інтелектуальної системи, яка б інтегрувала різні методи аналізу, забезпечувала високу точність, швидкість та масштабованість, а також надавала розгорнуту аналітику в процесі роботи.

1.8.1 Системні вимоги

Розроблена система створена у вигляді самостійного веб-застосунку з використанням мови програмування Java та сучасних технологій: Spring Boot для серверної частини, TensorFlow Java для нейронних мереж, OpenCV для обробки зображень, Hibernate для роботи з базою даних, PostgreSQL для збереження даних.

1.8.2 Опис функцій програмного забезпечення

Головними функціями системи мають бути надійне зберігання зображень продуктів та результатів їх аналізу з забезпеченням доступу до інформації з необхідною швідкістю. Система повинна мати можливість формувати власний набір даних для навчання моделей, а також розробляти і навчати нейронну мережу для класифікації якості продуктів. Інтеграція цієї мережі у серверну частину на Spring Boot з реалізацією REST API для обробки клієнтських запитів є ключовою задачаю в рамках розробки. Система повинна також забезпечувати веб-інтерфейс, що дозволяє завантажувати зображення і зручно переглядати результати аналізу.

Крім того, важливо мати шкалу якості, яка відображатиме ступінь свіжості продуктів – від неспілого до зіпсованого стану, щоб користувачі могли легко інтерпретувати отримані дані. Адміністративна панель має дозволяти моніторити якість продукції в режимі реального часу на виробничих лініях. Також необхідна перевірка точності моделей через графіки для оцінки їх ефективності за часом і надавати можливість додавати бібліотеки для порівняльного аналізу моделей. Не менш важливо, щоб інтерфейс включав сторінки для перевірки точності моделей з вибором випадкових зображень, контролю якості та налаштувань моделей із можливістю збереження параметрів у базі даних.

2 АНАЛІЗ ВИКОРИСТОВУВАНИХ ТЕХНОЛОГІЙ

2.1 Огляд засобів розробки

Успішна реалізація інтелектуальної системи аналізу якості харчових продуктів за зображенням залежить не лише від вибору основних технологій, але й від ефективного використання різноманітних інструментів розробки. Ці інструменти надають можливість підтримки ПО на всіх етапах розробки, починаючи реалізацією логіки і до його управління та розгортання готового продукту на сервері.

В якості мови серверної частини було обрано мову програмування Java завдяки її можливості роботи на різних типах пристроїв, великий вибір бібліотек і сумісність із такими фреймворками, як Spring Boot, Hibernate та TensorFlow Java. Це робить її придатною як для обробки логіки застосунків, так і для впровадження моделей ML [13].

Основним середовищем розробки є IntelliJ IDEA, що забезпечує глибоку інтеграцію з Java-технологіями, підтримує автодоповнення, налагодження та управління залежностями, що сприяє зменшенню кількості можливих помилок.

Spring Boot – основний фреймворк для побудови внутрішніх сервісів RESTful API. Завдяки його готовим модулям для веб-сервісів спрощується процес розробки програм [14].

Інтелектуальні модулі обробки та аналізу зображень використовують TensorFlow Java для запуску навчених моделей нейронних мереж безпосередньо в середовищі Java. Крім того, OpenCV використовується для завдань попередньої обробки зображень, таких як фільтрація, зміна розміру та виділення ознак, які є важливими для підвищення точності класифікації.

Для управління базами даних система використовує Hibernate ORM як рівень абстракції для роботи з SQL Server 2019, обраною системою

управління реляційними базами даних. Ця комбінація дозволяє ефективно зберігати зображення, інформацію про користувача та результати аналізу.

Інтерфейс користувача розроблений з використанням стандартних веб-технологій – HTML, CSS та JavaScript з бібліотекою ReactJS для створення динамічного та адаптивного користувацького інтерфейсу [15].

ReactJS – JavaScript-бібліотека, яка дозволяє розробникам створювати динамічні та адаптивні інтерфейси, що є критично важливим для сучасних застосунків різного типу складності [16].

Postman полегшує тестування та налагодження кінцевих точок REST, в той час як Swagger автоматично створює інтерактивну документацію API для полегшення співпраці та тестування. Для розгортання середовища використовується Docker, що контейнеризує компоненти для легкого управління залежностями на різних типах пристроїв.

2.2 Технології реалізації Backend частини застосунку

2.2.1 Середовище розробки IntelliJ IDEA

Одним із ключових інструментів, що використовується для розробки серверної частини інтелектуальної системи, є вбудоване в текстовий редактор середовище розробки IntelliJ IDEA.

Сучасні можливості, зокрема підказки, автоматичне завершення коду, перевірка синтаксису в реальному часі та вбудовані інструменти аналізу коду сприяють підвищенню ефективності розробки та зменшенню кількості помилок. Особливо цінною є глибока інтеграція з фреймворком Spring Boot, яка дозволяє розробникам легко створювати RESTful API, налаштовувати конфігураційні файли та взаємодіяти з залежностями проєкту.

Інтерфейс IntelliJ IDEA, що зображений на рисунку 2.1, орієнтований на зручність розробника, оскільки забезпечує чисту та інтуїтивно зрозумілу організацію робочого простору.

Середовище також підтримує роботу з системами контролю версій, зокрема Git, що забезпечує зручне відстеження змін, створення гілок та інтеграцію з віддаленими репозиторіями безпосередньо з IDE.

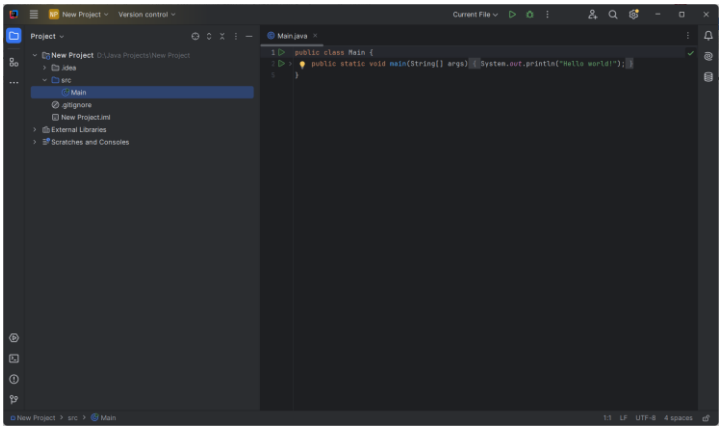

Рисунок 2.1 – Користувацький інтерфейс IntelliJ IDEA

Наявність інструментів для швидкої взаємодії з БД, відображенням структури проєкту та зручним управлінням збірками (через Maven або Gradle) робить IntelliJ IDEA універсальним середовищем для повного циклу розробки програмного забезпечення [17].

2.2.2 Мова програмування Java

Java – одна з найпоширеніших мов програмування, що вже понад два десятиліття залишається стандартом у розробці програмного забезпечення.

Java – приклад мови, яка використовує сувору типізацію, що сприяє створенню чіткої, структурованої архітектури проєкту. Завдяки віртуальній машині Java (JVM) розроблені програмні рішення можуть працювати незалежно від пристрою користувача.

Також ця мова надає доступ до необмеженої кількості готових бібліотек та фреймворків, таких як Spring Boot, Hibernate, TensorFlow Java, OpenCV та інші. Також варто відзначити багату інфраструктуру супровідних

інструментів – систем керування проєктами (Maven, Gradle), систем логування, тестування, профілювання та моніторингу продуктивності, що дозволяє розробляти продуктивні системи [18].

2.2.3 Фреймворк Spring Boot

Spring Boot – фреймворк, що суттєво спрощує створення та розгортання сервісів. Spring Boot являє собою надбудовою для фреймворку Spring Framework та пропонує автоматичну конфігурацію, вбудований сервер і набір готових компонентів для розробки веб-сервісів, RESTful API, доступу до баз даних, безпеки та інших функцій.

Однією з ключових переваг Spring Boot є принцип, який дозволяє мінімізувати необхідність налаштувань під кожного розробника. Це пришвидшує початок роботи над проєктом, дозволяючи зосередитися безпосередньо на бізнес-логіці замість конфігураційних файлів [19].

2.2.4 ORM Hibernate та СУБД SQL Server 2019

Hibernate використовується для об'єктно-реляційного відображення, та слугує як міст між Java-застосунком і реляційною базою даних, завдяки чому розробники можуть виконувати різноманітні операції з об'єктами бази даних, як зі звичайними об'єктами Java, спрощуючи збереження та пошук даних шляхом автоматичного створення запитів. Hibernate використовується для відображення моделей даних застосунку, а саме користувачів, зображень, результатів аналізу та конфігурацій моделей у відповідні таблиці в БД [20].

В якості СУБД обрана SQL Server 2019, яка є потужним рішенням для зберігання великої кількості інформації з високою продуктивністю та надійністю. SQL Server обирають за його продуктивність, функції безпеки та надійну підтримку складних запитів і транзакцій. Завдяки сумісності з Hibernate, SQL Server дозволяє ефективно організувати роботу з даними [21].

2.2.5 Бібліотека Lombok

Lombok – це бібліотека Java, яка допомагає скоротити шаблонний код, автоматично відтворючи часто використовувані методи, конструктори та інш. При розробці інтелектуальної системи Lombok впорядковує кодову базу, дозволяючи зосередитися на бізнес-логіці, а не на повторюваному коді. Використовуючи анотації, Lombok покращує читабельність коду та зручність супроводу, роблячи процес розробки швидшим та менш схильним до помилок. Це особливо важливо в складних застосунках, де задіяні численні моделі даних і сутності, оскільки допомагає зберегти код стислим і зрозумілим. Крім того, бібліотека добре підтримується сучасними IDE, зокрема IntelliJ IDEA, що забезпечує зручну роботу без необхідності додаткових налаштувань [22].

2.3 Технології для реалізації інтелектуального модуля

Основною метою застосунку є розпізнавання і класифікація зображень продуктів для оцінки їх свіжості та якості. Для цього застосовуються спеціалізовані бібліотеки, що дозволяють ефективно запускати моделі нейронних мереж і виконувати складні операції з обробки зображень.

2.3.1 TensorFlow Java

TensorFlow Java – це офіційний API для TensorFlow, фреймворку ML та є розробкою Google. Він надає можливість інтегрувати попередньо навчені моделі глибокого навчання безпосередньо в застосунок, що забезпечує високу продуктивність та швидкість обробки даних. TensorFlow Java підтримує широкий спектр операцій, що є основою нейронних мереж, та дозволяє запускати складні моделі для класифікації, регресії й інших завдань.

TensorFlow Java є важливою складовою інтелектуального модуля,

оскільки дозволяє реалізувати розпізнавання ознак якості продуктів на основі аналізу зображень у режимі реального часу, забезпечуючи гнучкість у виборі моделей та підвищує точність системи та дозволяє адаптувати її до різних типів продуктів і умов зйомки [23].

2.3.2 OpenCV

OpenCV (Open Source Computer Vision Library) – набір функцій для роботи з технологією комп'ютерного зору, що використовується для виконання операцій із зображеннями.

Ця бібліотека забезпечує широкий спектр функцій, таких як зчитування і збереження зображень, фільтрація шумів, зміна розміру, перетворення кольорового простору, підвищення контрастності, бінаризація, виявлення контурів, а також виділення ключових ознак. Усі ці операції дозволяють стандартизувати зображення, покращити якість вхідних даних та виділити характеристики, що безпосередньо впливають на точність класифікації.

Інтеграція OpenCV у Java-застосунок здійснюється через відповідні Java-обгортки, які дозволяють безпосередньо викликати функції бібліотеки у середовищі JVM. Поєднання OpenCV з TensorFlow Java створює ефективний програмний тандем, де OpenCV відповідає за підготовку зображень, а TensorFlow – за їх класифікацію за допомогою нейронних мереж [24].

2.4 Технології реалізації Frontend частини застосунку

2.4.1 Середовище розробки Visual Studio Code

Visual Studio Code (VS Code) є редактором коду, що набув популярності завдяки своїй легкості, гнучкості та розширюваності. Однією з головних переваг цього редактора є інтуїтивно зрозумілий інтерфейс, який сприяє зручній роботі з проєктом, а саме можливість одночасного відкриття

кількох файлів, інтегрований термінал, підтримка Git та система плагінів для автодоповнення коду, перевірки синтаксису та аналізу коду.

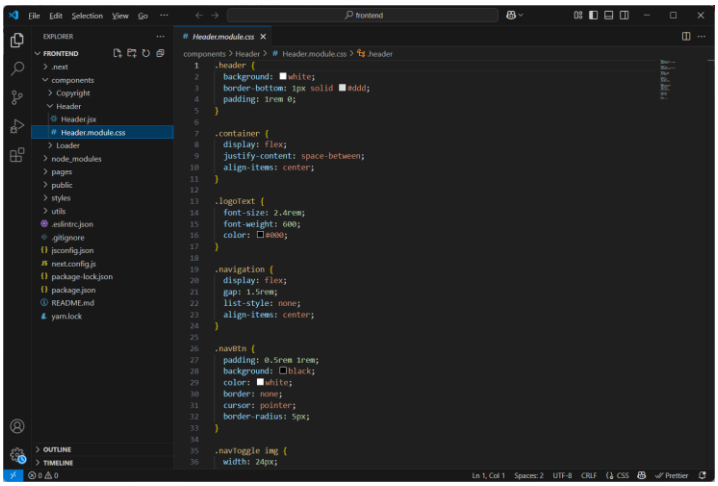

Рисунок 2.2 – Користувацький інтерфейс Visual Studio Code

Під час написання коду, було використано розширення: React Native Tools – для використання команд бібліотеки ReactJS і Prettier Code Formatter – для налаштування JS та JSX коду [25].

2.4.2 HTML, CSS, JavaScript

Основу клієнтської частини інтелектуальної системи становлять три фундаментальні технології – HTML, CSS та JavaScript, завдяки чому забезпечується цілісність відображення користувацького інтерфейсу.

Інтеграція HTML (HyperText Markup Language) необхідна для створення структури веб-сторінок. Саме за допомогою HTML визначаються елементи інтерфейсу: кнопки, поля введення, заголовки, таблиці, розділи тощо. HTML є основою, на якій побудована візуальна частина застосунку.

CSS (Cascading Style Sheets) відповідає за візуальне оформлення HTML-елементів. За допомогою CSS задаються кольори, шрифти, розміри, розміщення елементів, а також адаптивність інтерфейсу під різні типи

пристроїв. В інтелектуальній системі CSS відповідає за вигляд і зручність використання інтерфейсу, що є критично важливим для взаємодії користувача із застосунком.

JavaScript – це високорівнева, динамічна мова програмування, яка є невід'ємною частиною сучасної веб-розробки. Завдяки цій мові забезпечується швидка взаємодія з блоками інтерфейсу: отримання інформації про події, виконання запитів до БД. Використання HTML, CSS і JavaScript дозволяє створити сучасний, зручний та функціональний інтерфейс для користувачів [26].

2.4.3 Бібліотка ReactJS

ReactJS – це сучасна JavaScript-бібліотека, яка використовується для адаптивних користувацьких інтерфейсів. У межах інтелектуальної системи аналізу якості харчових продуктів за зображенням ReactJS дозволяє реалізувати гнучкий та зручний інтерфейс, що динамічно оновлюється без необхідності повного перезавантаження сторінки.

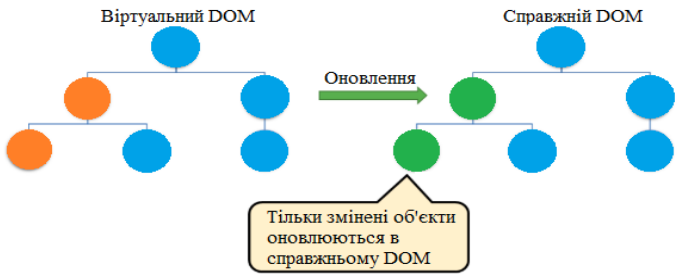

Рисунок 2.3 – Візуалізація роботи віртуального DOM

Бібліотека також підтримує інтеграцію маршрутизації через React Router, що забезпечує плавну навігацію, між сторінками перевірки точності системи, панелі адміністратора та інтерфейсом конфігурації моделі.

У поєднанні з такими технологіями, як HTML, CSS, JavaScript, та

RESTful API, що надається серверною частиною, ReactJS виступає потужним інструментом для реалізації сучасного веб-інтерфейсу, який адаптується до потреб користувачів і забезпечує взаємодію з функціоналом системи [27].

2.5 Додаткові технології для розробки та розгортання за стосунку

2.5.1 Інструменти для тестування API: Postman та Swagger

Для забезпечення якості та коректної роботи RESTful API, що реалізує серверна частина інтелектуальної системи, використовуються інструменти Postman та Swagger. Postman надає зручний інтерфейс, який дозволяє розробникам самостійно проводити тестуваня HTTP-запитів. За допомогою цього інструмента можна легко налаштувати параметри, заголовки та вміст запитів, що дозволяє ретельно перевіряти завантаження зображень, отримання результатів класифікації та API для керування моделями [28].

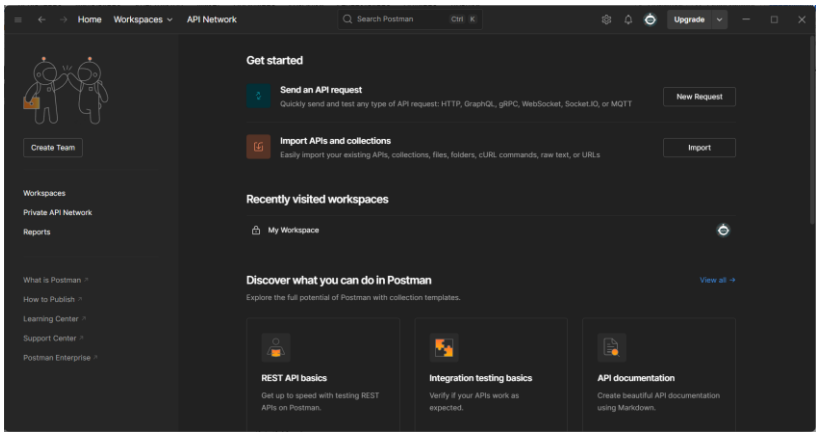

Рисунок 2.4 – Користувацький інтерфейс Postman

Swagger, у свою чергу, забезпечує автоматизоване створення інтерактивної документації API, що робить процес ознайомлення зі специфікацією сервісів прозорим і зручним для розробників. За допомогою Swagger UI користувачі можуть візуально переглядати доступні методи,

параметри запитів і типи відповідей, а також здійснювати тестові виклики безпосередньо із вікна браузера [29].

2.5.2 Docker

Дана технологія дозволяє упакувати програмні модулі разом із усіма залежностями в ізольовані контейнери, які гарантовано працюватимуть однаково у будь-якому середовищі – від локального комп’ютера розробника до серверів або хмарних платформ.

Застосування Docker значно спрощує управління залежностями, конфігураціями та масштабуванням додатку, що особливо важливо для складних систем із багатьма компонентами, як у випадку розроблюваної інтелектуальної платформи. Завдяки контейнеризації можна легко запускати окремі сервіси, наприклад, веб-сервер, базу даних або модулі машинного навчання, без ризику конфліктів у програмному середовищі [30].

2.5.3 Git

Для контролю версій та організації спільної роботи над проєктом використовується система керування версіями Git – розподілена система, яка дозволяє зберігати історію змін у коді, відновлювати попередні версії, а також паралельно працювати над різними гілками розробки без ризику втрати даних. Технологія дозволяє розгалужувати, об'єднувати та підтримувати різні версії програми, що дуже важливо при впровадженні нових функцій, виправленні помилок.

Інтеграція Git з популярними сервісами репозиторіїв, такими як GitHub, GitLab або Bitbucket, надає додаткові інструменти для перегляду змін, ведення обговорень та управління задачами [31].

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Огляд архітектури програми

Інтелектуальна система для аналізу якості харчових продуктів за зображенням побудована з використанням багаторівневої модульної архітектури, яка розділяє завдання між інтерфейсом користувача, логікою програми, зберіганням даних і компонентами машинного навчання.

На найвищому рівні система представляє собою три основних рівня: клієнтська та серверна частини, а також інтелектуальний модуль. Ці рівні взаємодіють через RESTful API, використовуючи JSON для обміну даними.

Клієнтська частина реалізована як односторінковий застосунок, який має зручний інтерфейс для завантаження зображень продуктів харчування, перегляду результатів аналізу, управління моделями та моніторингу параметрів точності. Для навігації між сторінками використовується React Router, а асинхронні виклики API дозволяють динамічно отримувати дані та взаємодіяти з користувачем без перезавантаження сторінок.

Серверна частина побудована за допомогою Spring Boot та виступає в ролі основного двигуна логіки застосунку. Серверна частина використовується для обробки HTTP-запитів, зображень, зберігання та отримання даних з бази даних SQL Server 2019 за допомогою Hibernate, а також контролює доступ за допомогою Spring Security. А також слугує мостом до інтелектуального модуля, який виконує аналіз зображень.

Інтелектуальний модуль, що був створений за допомогою Java, бібліотек TensorFlow Java та OpenCV, відповідає за завантаження моделей машинного навчання, попередню обробку вхідних зображень та повернення результатів прогнозування в реальному часі після завантаження зображень.

Уся система розроблена з урахуванням мікросервісного мислення, хоча в поточній версії була розгорнута як монолітне програмне рішення. Docker

використовується для контейнеризації коду застосунку, бази даних і клієнтської частини для розробки та інтеграції на сервер. Swagger використовується для документування API, Postman для тестування. Git забезпечує контроль версій та історію проєкту.

3.2 Реалізація серверної частини веб-застосунку

Серверна частина інтелектуальної системи аналізу якості харчових продуктів реалізована за допомогою Spring Boot і побудована за шаблоном багаторівневої архітектури, що сприяє розподілу завдань, легкості обслуговування та масштабованості (рисунок 3.1).

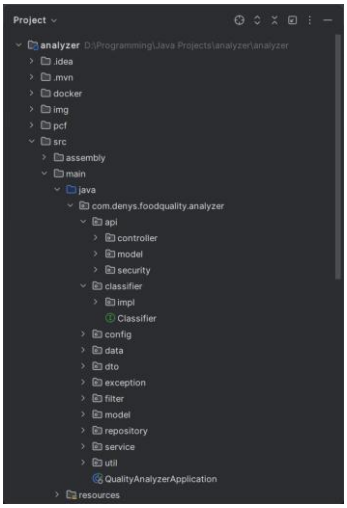

Рисунок 3.1 – Файлова структура серверної частини

Початковою точкою серверної частини є набір контролерів, кожен з яких прив'язаний до певних кінцевих точок для завантаження зображень, запитів на аналіз, конфігурації моделі та адміністративного моніторингу. Ці контролери делегують бізнес-логіку спеціальним класам, які інкапсулюють основну функціональність програми.

Клас ImageProcessingService, наведений на лістингу 3.1, відповідає за обробку вхідних зображень та здійснює їх попередню обробку, викликає

класифікатор на основі TensorFlow, а також формує результат аналізу у вигляді об’єкта RecognitionResult, що містить список розпізнаних об’єктів та зображення попереднього перегляду.

Лістинг 3.1 – Клас ImageProcessingService

```
      public class ImageProcessingService {
          private    final    Classifier<BufferedImage,    List<Recognition>>
classifier;
          private final int previewSize;
          @Autowired
          public ImageProcessingService(MobilenetV2Classifier classifier,

@Value("${tensorboot.maxExecutorsCount}") int maxExecutorsCount,
                                        @Value("${tensorboot.previewSize}")
int previewSize
          ) {
              this.classifier     =     new     PooledClassifier<>(classifier,
maxExecutorsCount);
              this.previewSize = previewSize;
          }
```

Клас TensorBootRestController, представлений на лістингу 3.2, реалізує REST-контролер, який обробляє вхідні HTTP-запити, пов’язані з обробкою зображень. В процесі роботи класу виконується перевірка вхідних даних та виконується перенаправлення обробки зображень сервісному рівню через ImageProcessingService.

Лістинг 3.2 – Вміст файлу TensorBootRestController

```
      public class TensorBootRestController {
          private final ImageProcessingService imageProcessingService;
          @Autowired
          public               TensorBootRestController(ImageProcessingService
imageProcessingService) {
              this.imageProcessingService = imageProcessingService;
          }
```

Контролер TensorBootController, наведений у лістингу 3.3, реалізує основну веб-інтерфейсну логіку для користувацької взаємодії із системою та відповідає за обробку HTTP-запитів, пов'язаних із завантаженням зображень, відображенням результатів.

Лістинг 3.3 – Код для контролеру TensorBootController

```
public class TensorBootController {
    private static final String IMAGE_ATTR = "IMAGE";
    private final ImageProcessingService imageProcessingService;
    @Autowired
    public TensorBootController(ImageProcessingService
imageProcessingService) {this.imageProcessingService =
imageProcessingService;}
    @RequestMapping(value = "/uploadForm", method = RequestMethod.GET)
    public String handleGetForm() {
        return "uploadForm";}
    @RequestMapping(value = "/img", method = RequestMethod.GET)
    public void handleGetImg(HttpServletRequest httpServletRequest,
HttpServletResponse httpServletResponse) throws IOException {
        BufferedImage image = (BufferedImage)
httpServletRequest.getSession().getAttribute(IMAGE_ATTR);
if (image != null) { httpServletResponse.setContentType("image/png");
            ImageIO.write(image, "png",
httpServletResponse.getOutputStream());} else {
httpServletResponse.sendError(HttpServletResponse.SC_NOT_FOUND);}}}
```

Для передачі результатів аналізу зображень між сервісами та контролерами в системі використовуються допоміжні класи. Клас Recognition (лістинг 3.4) представляє окремий об’єкт, розпізнаний на зображенні.

Лістинг 3.4 – Код для класу Recognition

```
public class Recognition {
    private String name; private Float confidence;}
```

Клас RecognitionResult (лістинг 3.5) є обгорткою над результатами обробки одного зображення.

Лістинг 3.5 – Код для класу RecognitionResult

```
public class RecognitionResult {
    private BufferedImage imagePreview; private List<Recognition>
recognitions;}
```

Для перевірки коректної роботи REST-контролера було реалізовано інтеграційний тест, представлений у лістингу 3.6. У цьому тесті імітується

HTTP-запит до кінцевої точки /TensorApi/recognizeFile, під час якого надсилається зображення банана у форматі JPEG. Очікується, що у відповіді буде повернено об’єкт із назвою «banana, fresh» та відповідним рівнем впевненості, що підтверджує здатність системи розпізнавати не лише тип продукту, а й оцінювати його якість.

Лістинг 3.6 – Код для інтеграційного тесту

```
public class TensorBootRestControllerIT {
    private static final String TEST_IMAGE_RESOURCE = "/banana.jpg";
    @Autowired
    private MockMvc mockMvc;
    @Autowired
    private MDCFilter mdcFilter;
    private byte[] testImage;
    @Before
    public void setup() throws IOException {
        mdcFilter.init(null);
        testImage =
IOUtils.toByteArray(TensorBootRestControllerIT.class.getResourceAsStream(TEST
_IMAGE_RESOURCE));
    }
}
```

Управління збереженням даних здійснюється за допомогою Hibernate ORM у поєднанні з SQL Server. Сутності, такі як LocalUser, а інтерфейси репозиторіїв забезпечують абстракцію над SQL-запитами. Таке налаштування дозволяє легко маніпулювати постійними даними без написання низькорівневого коду БД.

На лістингу 3.7 показано приклад сутності LocalUser, яка відображає користувача у базі даних за допомогою Hibernate ORM.

Лістинг 3.7 – Клас сутності LocalUser

```
public class LocalUser implements UserDetails {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private Long id;
  @Column(nullable = false, unique = true)
  private String username;
  @JsonIgnore
  @Column(nullable = false, length = 1000)
```

```
        private String password;
        @Column(nullable = false, unique = true, length = 320)
        private String email;
        private String firstName;
        private String lastName;
        @JsonIgnore
        @OneToMany(mappedBy = "user", cascade = CascadeType.ALL, orphanRemoval
= true)

        private List<VerificationToken> verificationTokens = new
ArrayList<>();
      }
```

На лістингу 3.8 наведено приклад інтерфейсу LocalUserRepository, який реалізує шаблон доступу до даних для сутності LocalUser. LocalUserRepository розширює інтерфейс ListCrudRepository з пакету Spring Data, що надає стандартні методи для виконання CRUD-операцій без необхідності самостійного написання SQL-запитів.

Лістинг 3.8 – Код файлу LocalUserRepository

```
      public          interface          LocalUserRepository          extends
ListCrudRepository<LocalUser, Long> {
        Optional<LocalUser> findByUsernameIgnoreCase(String username);
        Optional<LocalUser> findByEmailIgnoreCase(String email);}
```

Додатково, інтерфейс містить методи findByUsernameIgnoreCase та findByEmailIgnoreCase, що дозволяють знаходити користувачів за унікальним ідентифікатором користувача незалежно від регістру символів. Завдяки підтримці Spring Data JPA, реалізація цих методів створюється автоматично на основі їхніх імен.

Доступ до кінцевих точок захищено за допомогою Spring Security (лістинг 3.9), який підтримує автентифікацію користувачів та контроль доступу по рівнях доступу. Інструменти адміністрування, такі як перегляд показників продуктивності моделі або зміна конфігурацій, доступні лише користувачам з відповідними привілеями. Облікові дані і інформація про сеанси користувачів обробляються за допомогою зашифрованих токенів і контекстів безпеки, що підтримуються в HTTP-запитах.

Лістинг 3.9 – Вміст файлу WebSecurityConfig

```
public class WebSecurityConfig {
  private JWTRequestFilter jwtRequestFilter;
  public WebSecurityConfig(JWTRequestFilter jwtRequestFilter) {
  this.jwtRequestFilter = jwtRequestFilter;}
  public SecurityFilterChain filterChain(HttpSecurity http) throws
Exception { http.csrf().disable().cors().disable();
    http.addFilterBefore(jwtRequestFilter, AuthorizationFilter.class);
    http.authorizeHttpRequests()
        .requestMatchers("/product", "/auth/register", "/auth/login",
            "/auth/verify", "/auth/forgot", "/auth/reset", "/error",
"/websocket", "/websocket/**").permitAll().anyRequest().authenticated();
    return http.build();}}
```

На лістингу 3.10 наведено приклад використання бібліотек Lombok та MapStruct для спрощення розробки.

Лістинг 3.10 – Використання Lombook та MapStruct

```
public class UserDto {private String username; private String email;}
```

Код серверного застосунку був створений з можливістю розширення та впровадження інших моделей для роботи.

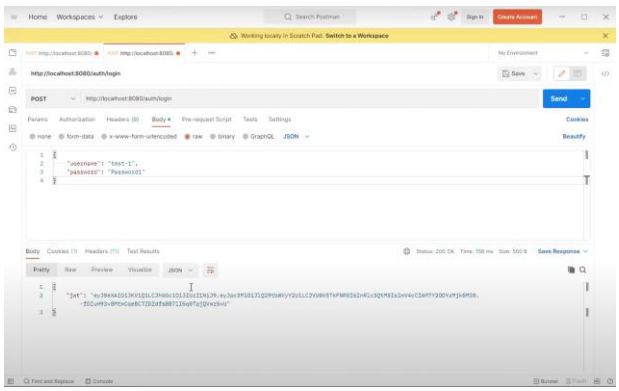

Рисунок 3.2 – Приклад тестування POST-запиту в Postman

Завдяки модульній організації сервісів і контролерів можна додавати нові моделі аналізу або кінцеві точки, не впливаючи на існуючу функціональність застосунку.

Код серверної частини також може бути інтегрований з інтелектуальним модулем, який викликається через сервісні виклики, що передають дані зображень і отримують результати прогнозування.

При розробці та тестуванні для перевірки та документування поведінки API використовуються такі інструменти, як Postman, які гарантують, легку взаємодію з сервером.

3.3 Реалізація інтелектуального модуля

Інтелектуальний модуль є основним аналітичним механізмом системи, оскільки в його роботі виконується оцінка якості харчових продуктів на основі аналізу зображень з використанням методів машинного навчання. Модуль побудований з використанням попередньо навченої моделі TensorFlow, налаштованої для задач класифікації зображень харчових продуктів. Модель класифікує вхідні зображення за попередньо визначеними категоріями якості, такими як «свіжі» або «зіпсовані».

Ядро класифікаційної логіки реалізовано в абстрактному класі AbstractClassifier на лістигну 3.11, який інкапсулює взаємодію з моделлю TensorFlow, а саме ініціалізацію графа, створення сесії та інш.

Лістинг 3.11 – Клас AbstractClassifier

```
      public abstract class AbstractClassifier<I, O, T> implements
Classifier<I, O> {
          private Graph model;
          private String inputLayerName;
          private String outputLayerName;
          public void init(byte[] graphBytes, String inputLayerName, String
outputLayerName) {
      Assert.notNull(graphBytes, "Model data shouldn't be null");
      Assert.notNull(inputLayerName, "Input layer name shouldn't be null");
      Assert.notNull(outputLayerName, "Output layer name shouldn't be null");
              model = new Graph();
              model.importGraphDef(graphBytes);
              this.inputLayerName = inputLayerName;
              this.outputLayerName = outputLayerName;
          }
      }
```

Щоб забезпечити необхідний рівень безпеки даних використовується обгортка PooledClassifier (лістинг 3.12), яка обмежує кількість одночасних потоків.

Лістинг 3.12 – Вміст файлу PooledClassifier

```
      public class PooledClassifier<I, O> implements Classifier<I, O> {
          private final Classifier<I, O> delegate;
          private final ExecutorService executorService;
          public     PooledClassifier(Classifier<I,     O>     delegate,     int
maxExecutorsCount) {this.delegate = delegate;
      executorService = Executors.newFixedThreadPool(maxExecutorsCount);}
          public O classify(I input) {
              try {
                  Future<O>     future     =     executorService.submit(()     ->
delegate.classify(input));
                  return future.get();
              } catch (InterruptedException | ExecutionException e) {
                  log.error("Error during task processing", e);
                  if (e.getCause() instanceof ServiceException) {
                      throw (ServiceException) e.getCause();
                  } else {
                      throw new ServiceException("Internal server error");
                  }}}}
```

Специфічна реалізація моделі базується на архітектурі MobileNetV2 та реалізована у класі MobilenetV2Classifier (лістинг 3.13), який успадковує AbstractClassifier і адаптує обробку вхідного зображення та результату для потреб аналізу харчових продуктів.

Лістинг 3.13 – Клас MobilenetV2Classifier

```
      public class MobilenetV2Classifier extends
AbstractClassifier<BufferedImage, List<Recognition>, Float> {
          private static final int BYTES_IN_MB = 1024 * 1024;
          private final ModelConfig modelConfig;
          private List<String> labels;
          @Autowired
          public MobilenetV2Classifier(ModelConfig modelConfig) {
              this.modelConfig = modelConfig;
          }
      }
```

Конфігурація моделі виконується за допомогою класу ModelConfig, який зчитує назви шарів моделі та шлях до графа з властивостей застосунку

(лістинг 3.14). Це дозволяє гнучко змінювати параметри моделі без необхідності модифікації коду.

Лістинг 3.14 – Конфігураційний файл

```
public class ModelConfig {

    @Value("${tensorboot.model.inputSize}")
    private Integer inputSize;
    @Value("${tensorboot.model.imageMean}")
    private Integer imageMean;
    @Value("${tensorboot.model.imageStd}")
    private Float imageStd;
    @Value("${tensorboot.model.inputLayerName}")
    private String inputLayerName;
    @Value("${tensorboot.model.outputLayerName}")
    private String outputLayerName;
    @Value("${tensorboot.model.path}")
    private String modelPath;
    @Value("${tensorboot.model.labelsResource}")
    private Resource labelsResource;

    @Value("${tensorboot.model.threshold}")
    private Float threshold;
}
```

Така архітектура дозволяє інтелектуальному модулю діяти як plug-and-play компонент. Завдяки механізму ін'єкції залежностей Spring, сервіси, пов'язані з моделлю, легко тестуються та підтримуються. Модульність також гарантує, що майбутні оновлення, такі як підтримка більшої кількості категорій продуктів харчування або інтеграція з відеоканалами в реальному часі – можуть бути досягнуті з мінімальними змінами в основному коді.

Більше того, прогнози, зроблені інтелектуальним модулем, реєструються разом з метаданими для підтримки постійної оцінки та підвищення точності моделі з часом. Ці показники зберігаються в структурованому форматі і візуалізуються на панелі адміністратора для моніторингу продуктивності моделі.

Інтеграція інтелектуального модуля не тільки автоматизує процес оцінки якості харчових продуктів, але й гарантує, що система залишається масштабованою, розширюваною та адаптованою до майбутніх досягнень у технології машинного навчання.

3.4 Реалізація клієнтської частини

Клієнтська частина системи розроблена як застосунок на React, що надає зручний інтерфейс для взаємодії з внутрішніми сервісами.

Веб-застосунок має модульну архітектуру з окремими компонентами для різних функціональних областей та використовує React Router для маршрутизації на стороні клієнта, що забезпечує плавний перехід між блоками застосунку без завантаження сторінки (лістинг 3.15).

Лістинг 3.15 – Приклад маршрутизації з React Router

```
import{BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import HomePage from './HomePage';
import UploadPage from './UploadPage';
function App() {
  return (
    <Router>
      <Routes>
        <Route path="/" element={<HomePage />} />
        <Route path="/upload" element={<UploadPage />} />
      </Routes>
    </Router>
  );
}
```

Управління станами на сторінці завантаження зображення здійснюється за допомогою вбудованих перехоплювачів станів React, що забезпечує швидку взаємодію з користувачем, а також контекстного API для глобального доступу до стану, де це необхідно (лістинг 3.16).

Лістинг 3.16 – Приклад використання перехоплювачів станів

```
export default function ExampleHooks() {
  const [dragging, setDragging] = useState(false);
  const [loading, setLoading] = useState(false);
  const [clientError, setClientError] = useState({ error: false,
message: "" });
  const dropzone = useRef(null);
  useEffect(() => {function handleDragOver(e) {
      e.preventDefault();
      e.stopPropagation();}
    dropzone.current.addEventListener("dragover", handleDragOver);
```

```
      return    ()    =>    {dropzone.current.removeEventListener("dragover",
handleDragOver);};}, []);
        return (
          <div ref={dropzone}>
            {dragging ? "Dragging!" : "Not dragging"}
            {loading && "Loading..."}
            {clientError.error && <p>Error: {clientError.message}</p>}
          </div>
        );
      }
```

Нижче наведено приклад реалізації такого компонента (лістинг 3.17), який відповідає за вибір, перевірку та відправлення файлу зображення на сервер для виконання операцій аналізу.

Лістинг 3.17 – Компонент для технології Drag&Drop

```
      function clickFileInput() {fileInputRef.current.click();}
        return (
          <>
            <div
              onDrop={handleDrop}
              onDragEnter={handleDragEnter}
              onDragLeave={handleDragLeave}
              onDragOver={(e) => e.preventDefault()}
              style={{
                border: dragging ? "2px dashed green" : "2px dashed gray",
                padding: "2rem",
                textAlign: "center",
                cursor: "pointer"
              }}
              onClick={clickFileInput}
            >
              {dragging ? "Drop your image here" : "Drag & drop an image or
click to upload"}
            </div>
            <input
              ref={fileInputRef}
              type="file"
              accept="image/*"
              style={{ display: "none" }}
              onChange={(e) => {
                if (e.target.files.length === 1) {
                  console.log("Uploading file:", e.target.files[0].name);
                }
              }}
            />
          </>
        );}
```

Сторінка завантаження зображення є основною частиною інтефейсу для взаємодії із системою перевірки якості продуктів харчування з

користувачами. Дана сторінка надає можливість користувачам легко та швидко завантажувати фотографії фруктів і овочів, які необхідно проаналізувати на предмет їхньої свіжості та загальної якості. Завантажені зображення автоматично надсилаються до API, де відбувається їх обробка за допомогою моделей машинного навчання.

Компонент Success, відповідає за відображення сторінки з підтвердженням успішного завантаження зображення та реалізований як функціональний React-компонент із використанням useRouter та useState.

Лістинг 3.18 – Код компоненту Success

```
const Success = () => {
  const router = useRouter();
  const { id, clientImg } = router.query;
  const [isCopied, setCopied] = useState(false);
  const freshness = 25;
  const handleCopyLinkClick = () => {
 navigator.clipboard.writeText(`https://image-uploader-
f08q.onrender.com/image/${id}`).then(() => {
      setCopied(true);
      setTimeout(() => setCopied(false), 4000);
});};
```

Сторінка тестування точності моделі призначена для адміністраторів і розробників, ця сторінка надає інструменти для тестування та оцінки точності різних моделей машинного навчання, інтегрованих в систему (лістинг 3.19). Користувачі можуть самостійно обирати моделі, виконувати тести на зразках зображень і переглядати детальні показники ефективності, представлені в таблицях і графіках.

Лістинг 3.19 – Приклад компонента для вибору моделі та запуску тесту

```
function ModelTesting() {
  const [selectedModel, setSelectedModel] = useState('');
  const [results, setResults] = useState(null);
  const handleTest = () => {
    fetch(`/api/test-model?model=${selectedModel}`)
      .then(res => res.json())
      .then(data => setResults(data));
  };
```

Наразі клієнтська частина перебуває в активній розробці і передбачає подальше розширення функціональності. У планах – інтеграція динамічних графіків для відстеження точності моделей машинного навчання у часі, що допоможе візуалізувати якість роботи системи. Також розробляється розширена адміністративна панель з можливістю керування налаштуваннями моделей, моніторингу активності користувачів і контролю якості продуктів на конвеєрі. Таке гнучке розширення інтерфейсу суттєво підвищить користувацький досвід і забезпечить ефективне управління якістю продуктів харчування.

3.5 Контейнеризація за допомогою Docker

Для забезпечення надійного та відтворюваної роботи інтелектуальної системи аналізу якості харчових продуктів, застосунок було контейнеризовано за допомогою Docker. Це дозволяє запускати сервіс в узгодженому середовищі незалежно від хост-системи.

Образ Docker базується на Ubuntu 16.04 і встановлює OpenJDK 21, як того вимагають Java-зв'язки TensorFlow. Файл .jar із Spring Boot та файл моделі TensorFlow копіюються до контейнера, а сервіс запускається за допомогою спеціального коду (лістинг 3.20).

Лістинг 3.20 – Вміст Dockerfile

```
      FROM ubuntu:16.04 RUN apt-get update && \ apt-get install -y openjdk-21-
jdk-headless ADD run.sh /usr/local/tensorboot/run.sh
      RUN chmod a+rx /usr/local/tensorboot/run.sh
      ADD                tensorboot-*.jar                /usr/local/tensorboot/
ADDmobilenet_v2_1.4_224_frozen.pb
/usr/local/tensorboot/model/mobilenet_v2_1.4_224_frozen.pb
      CMD cd /usr/local/tensorboot && ./run.sh EXPOSE 8080
```

Файл run.sh запускає веб-застосунок Spring Boot і забезпечує йому доступ до необхідного файлу моделі. Це забезпечує повну ізольованість середовищ та спрощує старт системи.

3.6 Документування REST API за допомогою Swagger

Для автоматичного ведення документації REST API використано інструмент Swagger (OpenAPI). Swagger дозволяє описати всі кінцеві точки, параметри запитів, формати відповідей, а також налаштувати інтерактивний веб-інтерфейс, де користувачі можуть ознайомитися з можливостями API та тестувати запити безпосередньо у браузері.

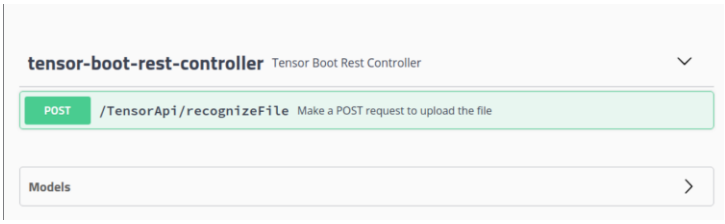

Рисунок 3.3 – Інтерфейс Swagger UI

Інтеграція Swagger у проєкт здійснюється через створення конфігураційного класу в серверній частині.

Лістинг 3.21 – Конфігураційний клас для Swagger

```
public class SwaggerConfig {
    public Docket swaggerDocket() {
        return new Docket(DocumentationType.SWAGGER_2)
             .useDefaultResponseMessages(false)
             .directModelSubstitute(LocalDate.class, String.class)
             .directModelSubstitute(LocalDateTime.class, String.class)
             .pathMapping("/")
             .select()
             .apis(RequestHandlerSelectors.any())
             .paths(Predicates.not(PathSelectors.regex("/error.*")))
             .build();}}
```

Swagger UI є зручним веб-застосунком з інтуїтивно зрозумілим інтерфейсом і дає змогу швидко ознайомитися з можливостями API, виконувати тестові запити і аналізувати результати без необхідності писати додатковий код.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Клієнтська частина веб-застосунку

Головна сторінка слугує точкою входу в інтелектуальну систему аналізу якості харчових продуктів за зображенням та надає чіткий і зручний інтерфейс, який знайомить користувачів із призначенням і можливостями застосунку, а також з варіантами навігації для доступу до всіх функцій.

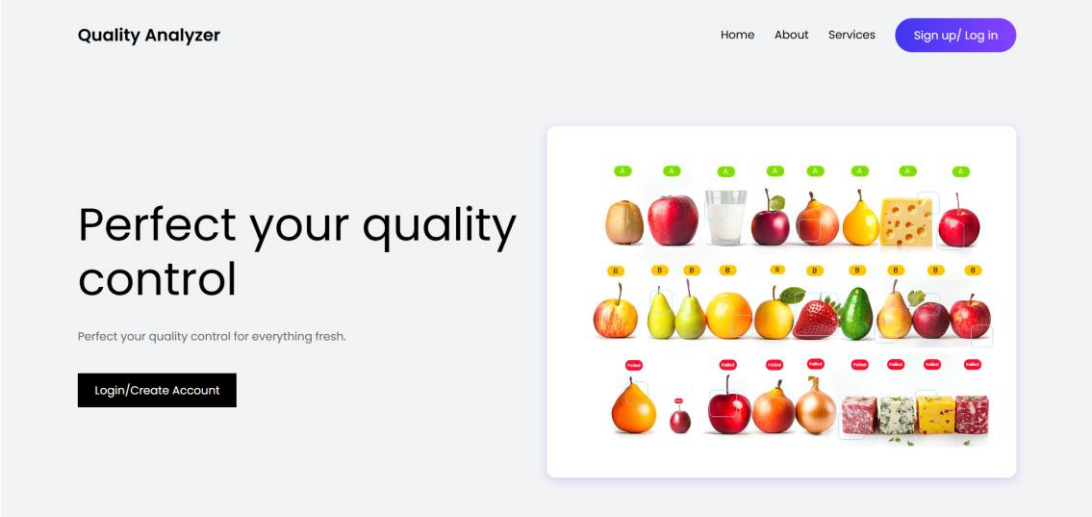

Рисунок 4.1 – Інтерфейс головної сторінки

Дизайн сторінки забезпечує просту взаємодію користувачів з усіма компонентами системи, дотримучись правил доступу до даних (рисунок 4.1).

4.2 Автентифікація у веб-застосунку

Для використання основних функцій, таких як перевірка точності моделей, конфігурація моделей і доступ до панелі адміністратора, необхідна автентифікація через сторінку «Sign up/Log in». Перейшовши, спочатку відкривається сторінка входу (рисунок 4.2), що дозволяє зареєстрованим

користувачам провести операцію авторизації в системі, використовуючи свою електронну пошту та пароль необхідний для входу в систему.

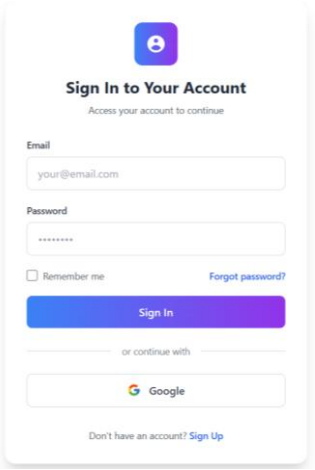

Рисунок 4.2 – Сторінка «Sign In»

Сторінка «Sign Up» необхідна для реєстрації нових користувачів (рисунок 4.3), якщо користувач ще не має аккаунт. Система перевіряє поля та забезпечує унікальність електронної пошти.

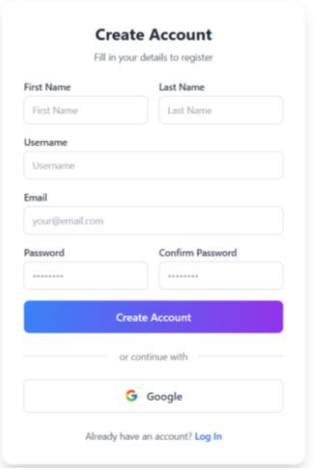

Рисунок 4.3 – Інтерфейс сторінки «Sign Up»

Після успішної реєстрації та входу в систему користувачі можуть перейти на сторінку профілю, використовуючи бічне меню зліва або своє ім'я

у верхньому правому куті панелі (рисунок 4.4). На сторінці відображаються дані про користувача, зібрані під час реєстрації.

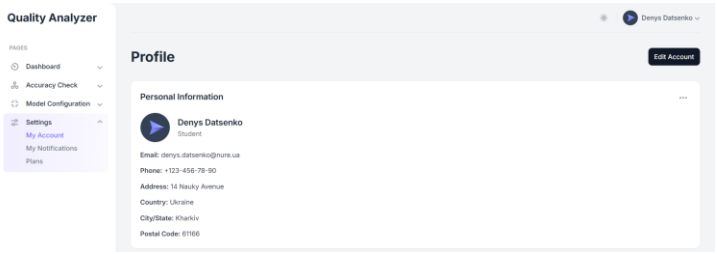

Рисунок 4.4 – Інтерфейс сторінки «Profile»

Дані користувача, що відображаються на сторінці профілю, надійно зберігаються і доступні лише авторизованому користувачеві.

4.3 Робота із адміністративною панеллю

Після входу користувач потрапляє на сторінку з основним інтерфейсом, доступним для авторизованих користувачів та слугує основною робочою зоною, де користувачі можуть завантажувати власні зображення харчових продуктів для аналізу системою.

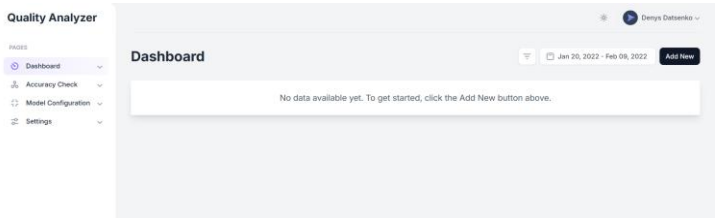

Рисунок 4.5 – Інтерфейс панелі нового користувача

На основі обраної моделі система прогнозує якість їжі на зображенні, класифікуючи її в одну із заздалегідь визначених категорій якості. Інтерфейс розроблений таким чином, щоб бути простим та інтуїтивно зрозумілим,

завдяки чому користувачі можуть оцінити якість продукції всього за кілька кроків та легко зрозуміти основні функції застосунку.

Щоб почати користуватися системою, користувач повинен завантажити своє перше зображення за допомогою кнопки «Add New». Сторінка завантаження зображення є ключовою для взаємодії користувачів із системою аналізу якості продуктів харчування (рисунок 4.6).

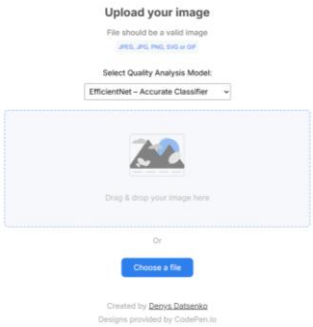

Рисунок 4.6 – Сторінка завантаження зображення

Інтерфейс надає змогу користувачам легко та швидко завантажувати фотографії фруктів і овочів, які необхідно проаналізувати на предмет їхньої свіжості та загальної якості.

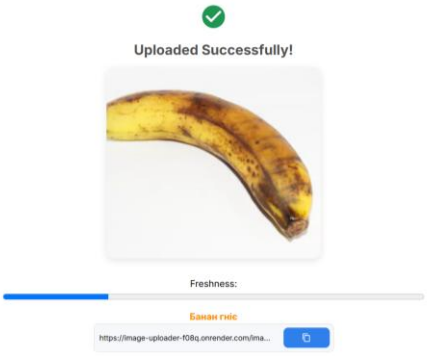

Рисунок 4.7 – Сторінка з результатом аналізу

Після відправлення зображення користувач отримує зворотній зв’язок у вигляді результатів аналізу (рисунок 4.7).

Коли панель містить дані (рисунок 4.8), то стає доступнем відображення структурованого та інформативного огляду усіх завантажених зображень разом з відповідними результатами аналізу. Кожен запис містить попередній перегляд зображення, обрану модель, рівень якості і оцінку точності моделі у відсотках.

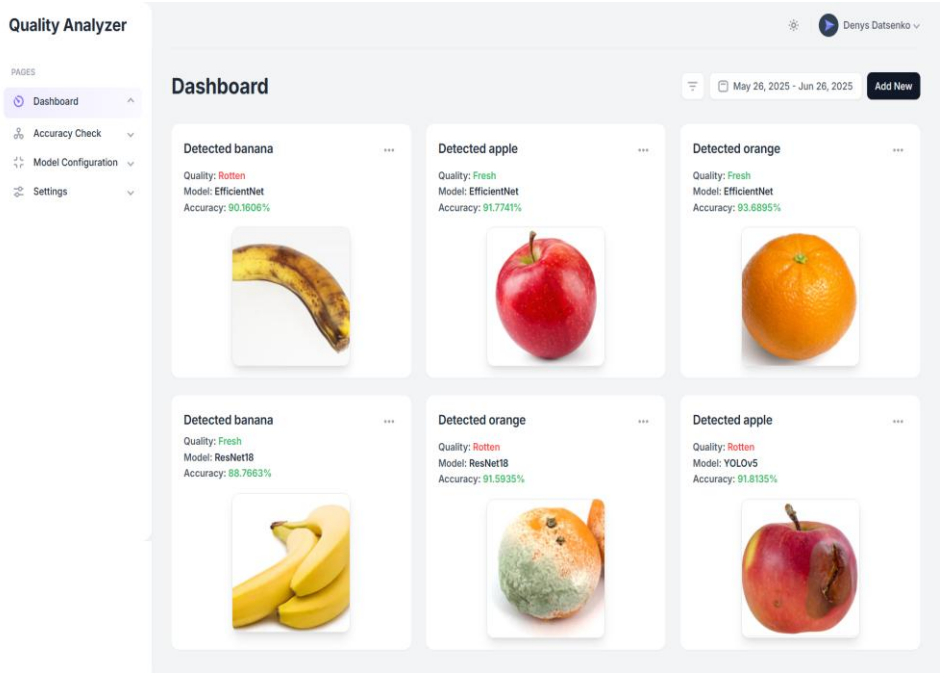

Рисунок 4.8 – Приклад користувацької панелі із зображеннями

Користувачі можуть швидко переглядати попередні результати, порівнювати результати різних моделей і, за бажанням, повторно запускати аналіз з використанням оновлених моделей. Інтерфейс також включає опції фільтрації та сортування, та надає можливість користувачам відсортувати результати за датою, рівнем достовірності або категорією якості. Завдяки адаптивному дизайну, панель зручно використовувати як на великих екранах, так і на мобільних пристроях.

Для проведення перевірки точності роботи та продуктивності моделей машинного навчання призначена сторінка «Accuracy Check» (рисунок 4.9), яка дозволяє користувачам вибирати доступні моделі і переглядати,

наскільки точно кожна з них спрацювала на реальних даних зображень, зібраних за попередні дні роботи.

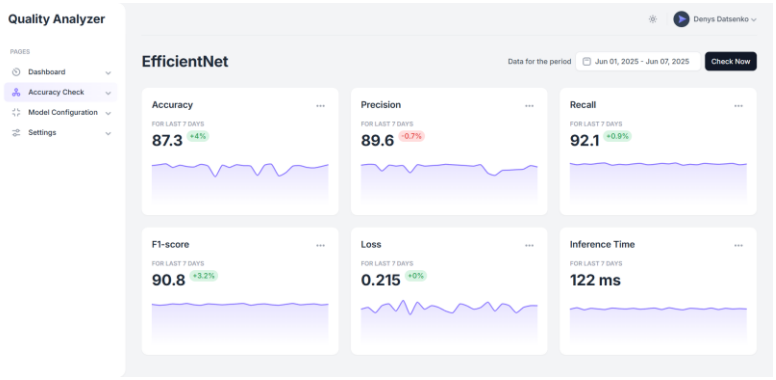

Рисунок 4.9 – Приклад користувацької панелі із даними

Метрики відіграють вирішальну роль в інтелектуальній системі аналізу якості харчових продуктів за зображенням, оскільки забезпечують об'єктивні показники для оцінки продуктивності, надійності та ефективності розгорнутих моделей машинного навчання.

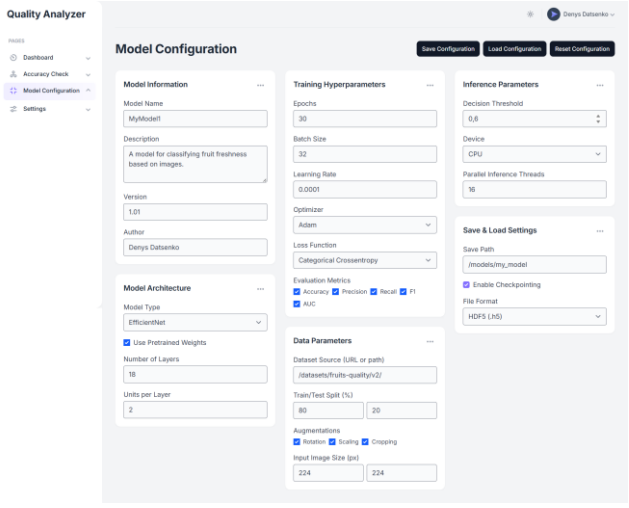

Рисунок 4.10 – Інтерфейс сторінки «Model Configuration»

Сторінка «Model Configuration» надає користувачам можливість керувати та налаштовувати моделі машинного навчання, що

використовуються для аналізу якості харчових продуктів за зображеннями (рисунок 4.10). Ця сторінка необхідна для підтримки оптимальної продуктивності моделі та адаптації системи до мінливих умов або нових наборів даних.

Сторінка складається з логічно структурованих блок-карток, які охоплюють всі ключові аспекти: загальну інформацію про модель, вибір архітектури, гіперпараметри тренування, параметри даних, збереження та інференс.

Для підвищення зручності, доступності та загального комфорту користувачів, особливо під час тривалої роботи з інтерфейсом, веб-застосунок включає перемикання теми відображення (рисунок 4.11).

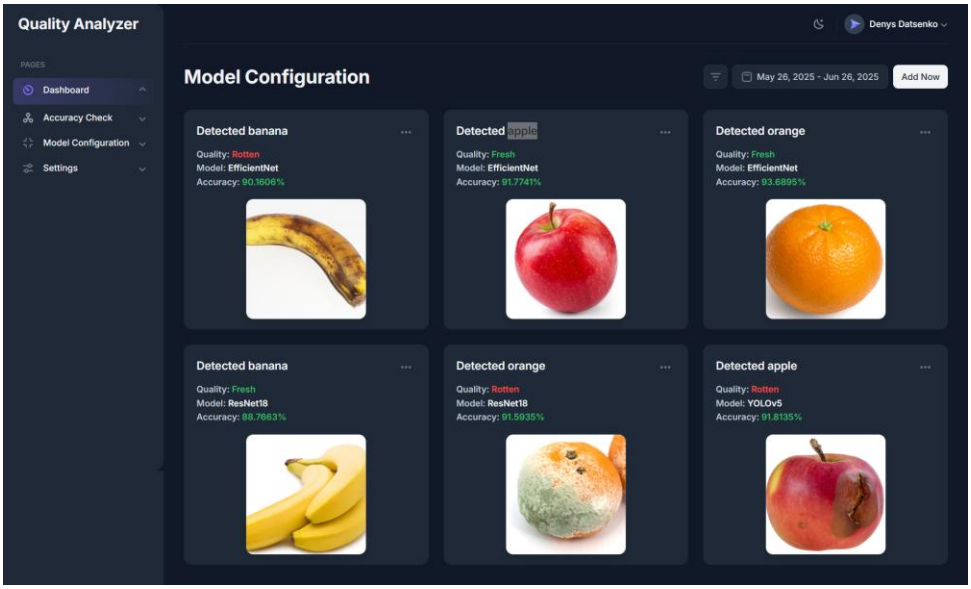

Рисунок 4.11 – Темна тема відображення

Користувачі можуть самостійно змінювати візуальне оформлення, обираючи між світлим і темним режимами. Такі можливості налаштування не лише знижують навантаження на очі, але й покращує загальне сприйняття інформації, забезпечуючи більш приємний і персоналізований досвід взаємодії з системою.

ВИСНОВКИ

Розвиток технологій комп’ютерного зору та глибокого навчання суттєво відкрив нові перспективи для автоматизованого контролю якості харчових продуктів, чим підвищив ефективність систем автоматизованого контролю якості харчових продуктів. Інтеграція нейронних мереж у веб-застосунки дає змогу здійснювати візуальну оцінку продукції у автоматичному режимі, виявляючи дефекти, ступінь свіжості та стиглості.

Проведений аналіз існуючих рішень показав, що на ринку ще відсутня універсальна система, здатна поєднати гнучкість налаштувань, розширюваність, підтримку локальних культур і доступність для малого та середнього бізнесу. Розроблена система прагне вирішити ці обмеження завдяки відкритій архітектурі та можливості навчання моделей під конкретні умови використання.

Запропонована інтелектуальна система аналізу якості за зображенням демонструє можливість ефективного поєднання технологій машинного навчання, сучасної веб-архітектури та обробки зображень для вирішення задачі оцінювання свіжості продукції. Веб-архітектура на основі сучасних фреймворків забезпечує масштабованість, гнучкість і зручність використання, дозволяючи інтегрувати нові моделі, покращувати аналітику та взаємодіяти з користувачами через інтуїтивний інтерфейс.

Системи такого типу змінюють парадигму контролю якості: від ручної оцінки до інтелектуального аналізу великих обсягів візуальних даних. Вже зараз існують приклади комерційних та наукових розробок, які демонструють практичну доцільність подібних рішень у промисловості.

Подальші дослідження зосереджуються на вдосконаленні моделей, розширенні наборів даних, підвищенні адаптивності до нових видів продукції та інтеграції з виробничими лініями. Це відкриває перспективу створення автономних платформ для контролю якості.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Food & Agriculture Organization. State of food and agriculture 2021. Food & Agriculture Organization of the United Nations, 2022

2. Quality Control in the Food Industry: A Guide. TDI Packsys. URL: https://www.tdipacksys.com/blog/quality-control-in-the-food-industry (дата звернення: 28.05.2025)

3. Brosnan T., Sun D.-W. Improving quality inspection of food products by computer vision. Journal of food engineering. 2004. С 3–16

4. Understanding Food Safety Regulations and Compliance. FoodReady AI. URL: https://foodready.ai/blog/food-safety-regulations-compliance (дата звернення: 28.05.2025)

5. Xing L., Wang Y. та інші. Application of computer vision technology in agricultural products and food inspection. Journal of physics: conference series. 2021. С 32–45

6. Kamilaris A., Prenafeta-Boldú F. X. Deep learning in agriculture: a survey. Computers and electronics in agriculture. 2018. С 70–90

7. A. Khan, A. Sohail, U. Zahoora та A. S. Qureshi. A survey of the recent architectures of deep convolutional neural networks. Artificial intelligence review. 2020. С 5455–5516

8. Rahnemoonfar M., Sheppard C. Deep Count: Fruit counting based on deep simulated learning. Sensors. 2017. С 905–912

9. Zhao X., Wang L. та інші. A review of convolutional neural networks in computer vision. Artificial intelligence review. 2024. С 71–72

10. Clarifresh: AI-Powered Quality Control for Fresh Produce. Clarifresh. URL: https://clarifresh.com (дата звернення: 28.05.2025)

11. AgroScout – Detection Done Right. AgroScout. URL: https://agroscout.com (дата звернення: 28.05.2025)

12. TOMRA: Transforming how we obtain, use and reuse our planet's resources. TOMRA. URL: https://www.tomra.com (дата звернення: 28.05.2025)

13. Introduction to Java. Geeks for geeks. 2025. URL: https://www.geeksforgeeks.org/introduction-to-java/ (дата звернення: 03.06.2025)

14. Spring Boot. Spring.io. URL: https://spring.io/projects/spring-boot (дата звернення: 03.06.2025)

15. Кантор І. Вступ до JavaScript. Сучасний підручник з JavaScript. 2025. URL: https://uk.javascript.info (дата звернення: 03.06.2025)

16. Meta Platforms, Inc. React – A JavaScript library for building user interfaces. React.dev. URL: https://react.dev/reference/react (дата звернення: 03.06.2025)

17. JetBrains S.R.O. IntelliJ IDEA | Features. JetBrains. URL: https://www.jetbrains.com/idea/features (дата звернення: 03.06.2025)

18. Oracle. Java Documentation – Get Started. Oracle Help Center. 2018. URL: https://docs.oracle.com/en/java (дата звернення: 04.06.2025)

19. VMware. Spring Boot Reference Documentation Overview. Spring.io. URL: https://docs.spring.io/spring-boot (дата звернення: 04.06.2025)

20. Red Hat. Hibernate ORM 7.0 Documentation. Hibernate ORM. URL: https://hibernate.org/orm/documentation/7.0 (дата звернення: 04.06.2025)

21. Microsoft Corp. SQL Server Technical Documentation. Microsoft Learn. URL: https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver17 (дата звернення: 04.06.2025)

22. The Project Lombok Authors. Project Lombok. Project Lombok. URL: https://projectlombok.org (дата звернення: 05.06.2025)

23. Google Brain Team. TensorFlow for Java. TensorFlow. URL: https://www.tensorflow.org/jvm (дата звернення: 05.06.2025)

24. Intel. OpenCV – Introduction. OpenCV Documentation. 2025. URL: https://docs.opencv.org/4.x/d1/dfb/intro.html (дата звернення: 05.06.2025)

25. Microsoft Corp. Documentation for Visual Studio Code. Visual Studio

Code. URL: https://code.visualstudio.com/docs (дата звернення: 05.06.2025)

26. Mozilla. Web Technology For Developers. MDN Web Docs. 2025. URL: https://developer.mozilla.org/en-US/docs (дата звернення: 05.06.2025)

27. Meta Platforms, Inc. Quick Start – React Learning. React.dev. URL: https://react.dev/learn (дата звернення: 05.06.2025)

28. Postman Inc. Postman Documentation Overview. Postman Learning. URL: https://learning.postman.com/docs (дата звернення: 06.06.2025)

29. SmartBear Software. API Documentation Tools. Swagger. URL: https://swagger.io/solutions/api-documentation (дата звернення: 06.06.2025)

30. Docker Inc. Docker Guides. Docker Documentation. 2025. URL: https://docs.docker.com/guides (дата звернення: 06.06.2025)

31. Pro Git Book. Git-SCM. 2025. URL: https://git-scm.com/book (дата звернення: 06.06.2025)

32. R. Johnson та інші. Professional Java Development with the Spring Framework. Wiley & Sons, Incorporated. 2011. С 676