Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Ком'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 ідентифікації ваги великої рогатої худоби
(тема)

Виконав:
здобувач ___2025___ року навчання,
групи _____КІУКІ-21-10_____
_____Андрій ЧЕРКАСОВ_____
(власне ім'я, прізвище)

Спеціальність 123 Комп'ютерна інженерія
_____
(код і повна назва спеціальності)
Тип програми ____освітньо-професійна_____
Освітня програма _____
_____Комп'ютерна інженерія_____
(повна назва освітньої програми)

Керівник ___доц. каф. КІТС Олег ІЛЮНІН___
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри _____ _____ ___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

Освітня програма Комп'ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«____»_____ 2025 р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Черкасову Андрію Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інтелектуальна система ідентифікації ваги великої рогатої худоби

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

2. Термін подання студентом роботи до екзаменаційної комісії 20.06.2025

3. Вхідні дані до роботи

1. Набір даних зображень великої рогатої худоби
2. Методи комп'ютерного зору та глибокого навчання
3. Архітектури нейронних мереж: UNet, ResNet18, EfficientNetB3.
4. Технології: Python, PyTorch, TensorFlow, Keras, Flask, ReactJS.
5. Наукові статті та огляди за темою роботи.

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналіз предметної області.
2. Аналіз використовуваних технологій.
3. Програмна реалізація моделей, клієнтської та серверних частин
4. Інструкція користувача
5. Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) _____

Слайд-презентація – 14 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	Термін виконання етапів роботи	Примітка
1	Затвердження дипломного проєкту	26.05.2025	Виконано
2	Аналіз предметної області та існуючих рішень	27.05.2025-28.05.2025	Виконано
3	Вибір технологій та інструментальних засобів	29.05.2025-30.05.2025	Виконано
4	Планування та проектування архітектури	30.05.2025-31.05.2025	Виконано
5	Розробка моделей машинного навчання	31.05.2025-04.06.2025	Виконано
6	Розробка серверної та клієнтських частин	04.06.2025-07.06.2025	Виконано
7	Інтеграція компонентів та тестування системи	08.06.2025-10.06.2025	Виконано
8	Написання пояснювальної записки до проекту	10.06.2025-17.06.2025	Виконано
9	Підготовка презентації	18.06.2025	Виконано
10	Захист кваліфікаційної роботи	25.06.2025	Виконано

Дата видачі завдання 26 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Ілюнін О.О.
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 60 с., 24 рис., 2 табл., 1 дод., 34 джерел.

ІНТЕЛЕКТУАЛЬНА СИСТЕМА, ІДЕНТИФІКАЦІЯ ВАГИ НЕЙРОННА МЕРЕЖА, АНАЛІЗ ЗОБРАЖЕНЬ, ШТУЧНИЙ ІНТЕЛЕКТ, PYTHON, СЕГМЕНТАЦІЯ ЗОБРАЖЕНЬ, АУГМЕНТАЦІЯ ДАНИХ, КОМП’ЮТЕРНИЙ ЗІР

Метою кваліфікаційної роботи є розробка інтелектуальної системи яка використовує комп'ютерні технології для автоматизованої обробки зображень великої рогатої худоби та оцінювання фізичних параметрів тварин для подальшої ідентифікації їх ваги.

Об'єктом дослідження є процеси оцінки фізичних параметрів великої рогатої худоби та їх вплив на економічну ефективність фермерських господарств.

Предметом дослідження є методи та моделі оцінки ваги тварин з використанням теорії нейронних мереж та методів комп'ютерного зору.

У роботі проведено аналіз зображень тварин за допомогою комп'ютерних технологій. Для цього використовуються згорткові нейронні мережі на базі моделей EfficientNetB3 та resnet18 для сегментації та формування ключових точок з зображень тварини для подальшої ідентифікації її характеристик.

Розроблена система дозволяє автоматизувати процес вимірювання ваги худоби, що значно знижує витрати на ручну працю та покращує економічну ефективність фермерських господарств. Імітаційне моделювання показало, що використання системи дозволяє отримати точніші оцінки ваги тварин в порівнянні з традиційними методами.

ABSTRACT

The explanatory note of the qualification work: 60 pages, 24 figures, 2 tables, 1 appendices, 34 references.

INTELLIGENT SYSTEM, WEIGHT IDENTIFICATION, NEURAL NETWORK, IMAGE ANALYSIS, ARTIFICIAL INTELLIGENCE, PYTHON, IMAGE SEGMENTATION, DATA AUGMENTATION, COMPUTER VISION

The aim of this qualification work is the development of an intelligent system for identifying the weight of cattle, which uses computer technologies for processing images of animals and evaluating their physical parameters. The system provides automated weight estimation of livestock based on images and information about their physical parameters.

The object of the research is the processes of evaluating the physical parameters of cattle and their impact on the economic efficiency of farm enterprises.

The subject of the research is methods and models of animal weight assessment using the theory of neural networks and computer vision methods..

This work conducts an image analysis of animals using computer technologies. For this, convolutional neural networks based on EfficientNetB3 and Resnet18 models models are used for segmentation and formation of key points from animal images for further identification of its characteristics.

The developed system allows for the automation of the livestock weight measurement process, significantly reducing labor costs and improving the economic efficiency of farm enterprises. Simulation modeling showed that using the system allows you to obtain more accurate estimates of animal weight compared to traditional methods.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API (Application Programming Interface) – Інтерфейс прикладного програмування

CNN (Convolutional Neural Network) – Згорткова нейронна мережа

COCO (Common Objects in Context) – Формат анотацій (використовувався для даних ключових точок)

CPU (Central Processing Unit) – Центральний процесор

CSV (Comma-Separated Values) – Формат файлів для зберігання табличних даних

DOM (Document Object Model) – Об'єктна модель документа (контекст React)

HTTP (Hypertext Transfer Protocol) – Протокол передачі гіпертексту

IDE (Integrated Development Environment) – Інтегроване середовище розробки

IoU (Intersection over Union) – Метрика для оцінки точності сегментації

JSON (JavaScript Object Notation) – Текстовий формат обміну даними

JWT (JSON Web Token) – Веб-токен JSON (для автентифікації)

MAE (Mean Absolute Error) – Середня абсолютна помилка

RMSE (Root Mean Square Error) – Середньоквадратична похибка

ML (Machine Learning) – Машинне навчання

MLP (Multi-Layer Perceptron) – Багатошаровий перцептрон

ORM (Object-Relational Mapping) – Об'єктно-реляційне відображення

PCK (Percentage of Correct Keypoints) – Відсоток коректних ключових точок.

PNG (Portable Network Graphics) – Растровий формат файлів

PyTorch – Бібліотека машинного навчання

ReactJS – JavaScript бібліотека для створення інтерфейсів користувача

ReLU (Rectified Linear Unit) – Функція активації в нейронних мережах

ResNet (Residual Network) – Архітектура згорткової нейронної мережі

REST (Representational State Transfer) – Архітектурний стиль взаємодії компонентів розподіленого застосунку в мережі

SQLite – Реляційна система керування базами даних

SPA (Single Page Application) – Односторінковий застосунок

TensorFlow – Бібліотека машинного навчання

U-Net – Архітектура згорткової нейронної мережі для сегментації

UI (User Interface) – Інтерфейс користувача

URL (Uniform Resource Locator) – Уніфікований покажчик ресурсу

VS Code (Visual Studio Code) – Редактор вихідного коду

WSGI (Web Server Gateway Interface) – Програмний інтерфейс шлюзу веб-сервера (контекст Flask)

JSX (JavaScript XML) – Синтаксичне розширення JavaScript

ВРХ (Велика рогата худоба) – Об'єкт дослідження

ІІС – Інформаційна інтелектуальна система

КЗ (Комп'ютерний зір) – Технологія обробки зображень методами штучного інтелекту

ВСТУП

У сучасному світі зростання кількості інформації та новітніх технологій відкриває широкі можливості для пошуку та аналізу даних.

Однією з галузей, що активно використовує ці технології, є сільське господарство. Завдяки розвитку комп'ютерних технологій, автоматизація процесів на фермах стає більш ефективною.

Зокрема, у процесі вирощування сільськогосподарських тварин контроль за параметрами, такими як жива вага, стає важливим інструментом для управління харчуванням і регламентним доглядом за тваринами.

Відхилення від нормальних значень живої ваги можуть свідчити про проблеми зі здоров'ям, умовами утримання або харчуванням, що прямо впливає на економічну ефективність господарства.

На даний момент більшість таких вимірів виконуються вручну, що є витратним і часто неефективним процесом. Зважаючи на це, застосування інтелектуальних алгоритмів для оцінки живої маси тварин є актуальним кроком до оптимізації сільськогосподарських процесів.

Метою цього дослідження є розробка ІІС із застосуванням нейронних мереж для оцінки живої ваги тварин, використовуючи фотограмметричні методи для визначення розмірів тіла тварини, що дозволить знизити витрати на обслуговування та підвищити ефективність сільськогосподарських процесів.

Для цього застосовано кілька нейромережевих моделей: згорткові нейронні мережі для сегментації тварини на фотографіях і визначення її ключових точок, а також згорткова нейронна мережа для оцінювання маси тварини на основі її фізичних параметрів, визначених за допомогою сегментованого зображення та ключових точок. Очікується, що такий підхід дозволить визначити розміри тіла тварини, знизити витрати на обслуговування та підвищити ефективність у сільськогосподарчому тваринництві.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд традиційних методів визначення ваги худоби

Інформація про точну вагу корови є базовим аспектом ефективного догляду за стадом великої рогатої худоби (ВРХ). Цей показник критично важливий для низки ключових господарських операцій. По-перше, від ваги тварини залежить правильне дозування ветеринарних препаратів та медикаментів. Неточна оцінка ваги може призвести до недостатньої дози ліків, що робить лікування неефективним, або до передозування, яке може бути токсичним для тварини. Дослідження показують, що недооцінка ваги є поширеним ризиком, що безпосередньо веде до занижених доз ліків [1, 2].

По-друге, вага є основою для розрахунку та балансування раціонів годівлі. Правильне харчування, адаптоване до ваги та фізіологічного стану корови, забезпечує оптимальний ріст молодняку, високу молочну продуктивність, добре здоров'я та репродуктивну функцію. Неправильна оцінка ваги може призвести до перегодовування або недогодовування, що негативно впливає як на продуктивність тварин, так і на економіку господарства через неефективне використання кормів [3].

По-третє, регулярний моніторинг ваги дозволяє відстежувати темпи росту ремонтного молодняку, оцінювати загальний стан здоров'я стада та своєчасно виявляти можливі проблеми. Досягнення цільових показників ваги у певні періоди життя тварини є критично важливим для досягнення виробничих та економічних цілей ферми. Крім того, вага є важливим фактором при прийнятті рішень щодо термінів осіменіння телиць та при визначенні вартості тварини під час продажу, особливо коли доступ до ваг обмежений [4].

Незважаючи на важливість точного зважування, використання спеціальних ваг для худоби часто є складним завданням для багатьох

господарств. Ваги можуть бути дорогими у придбанні та обслуговуванні, громіздкими, потребувати спеціально обладнаного місця та часу для зважування кожної тварини. Сам процес зважування викликає стрес у корів, що негативно позначається на їхньому самопочутті та продуктивності. Тому виникає нагальна потреба в надійних безконтактних методах оцінки ваги без використання ваг [3].

Важливо одразу наголосити, що спроба визначити вагу корови «на око» є вкрай ненадійним методом. Дослідження чітко демонструють, що візуальна оцінка часто призводить до значної недооцінки реальної ваги тварини [1]. Наприклад, в дослідженні [2] ветеринари недооцінювали вагу в 64.7% випадків (в середньому на 63 кг), а фермери – у 80.9% випадків (в середньому на 97 кг).

Така систематична помилка в бік заниження ваги створює пряму загрозу здоров'ю тварин через ризик неефективного лікування заниженими дозами препаратів. Це підкреслює необхідність використання будь-яких методів, що базуються на об'єктивних вимірюваннях, а не на суб'єктивній візуальній оцінці. Знання ваги – це не просто отримання числа, а ключ до інтегрованої системи управління фермою, що охоплює годівлю, нагляд за здоров'ям тварин, відтворення поголів'я та економіку господарств.

Один із найпростіших способів оцінити вагу великої рогатої худоби без використання ваг – застосування спеціальних вимірювальних стрічок, які іноді називають «ваговими стрічками» або «аніметрами» [5]. На відміну від звичайних сантиметрових стрічок, такі стрічки мають шкалу, що відображає приблизну вагу тварини в кілограмах або фунтах поряд із поділками довжини [6].

Принцип дії цих стрічок базується на вимірюванні обхвату грудної клітки корови в ділянці серця, одразу за лопатками. Саме цей обхват має високу кореляцію з живою масою тіла у великої рогатої худоби, що дозволяє доволі точно оцінити вагу на основі одного виміру. Для правильного використання стрічки її слід обережно обернути навколо грудної клітки в області серця так, щоб вона щільно прилягала до тіла, не перекручувалася і не була перетягнута.

Значення ваги зчитується безпосередньо зі шкали на стрічці відповідно до обхвату [1].

На ринку представлено різноманітні вагові стрічки, розраховані на різні види тварин – велику рогату худобу, свиней, коней, овець, кіз, оленів тощо. Існують стрічки, спеціально розроблені для ВРХ певних напрямів продуктивності (молочні чи м'ясні породи) або навіть конкретних порід, таких як голштинська, симентальська, гернзейська чи джерсейська, а також для різних вікових груп – телят, нетелів і дорослих корів. Деякі стрічки мають подвійну шкалу в кілограмах і фунтах, а також у сантиметрах і дюймах. Зазвичай їх виготовляють із міцних, гнучких та водостійких матеріалів, таких як ПВХ або скловолокно. Існують також окремі варіанти для телят, де вага визначається за обхватом п'ястка, а не грудей.

Хоча спеціальні стрічки є зручним і відносно точним інструментом, їхня ефективність значною мірою залежить від правильного підбору стрічки під тип, породу та вік тварини, а також від дотримання техніки вимірювання. Варто пам'ятати, що стрічки, відкалібровані для великих європейських порід, можуть істотно завищувати вагу місцевих або менших за розміром порід через морфологічні відмінності. Перед покупкою важливо уточнити, для яких саме порід або типів тварин призначена конкретна модель [3].

Незважаючи на те, що деякі виробники заявляють високу точність, наприклад, похибку близько 2,04% [7], незалежні дослідження вказують на можливість значно більших відхилень. Так, в одному дослідженні було встановлено, що вагові стрічки завищували вагу швіцької породи більш ніж на 21 кг, а джерсейської – понад на 44 кг у порівнянні з реальною вагою на вагах [3].

При цьому, за надійністю, вагові стрічки поступалися формулі Шеффера, але переважали стрічку Rondo та формулу Агарвала. Це свідчить про те, що отримані за допомогою стрічки результати варто сприймати як орієнтовні, особливо якщо стрічка не адаптована до конкретної породи.

Спеціальні вагові стрічки доступні у продажу через постачальників ветеринарних товарів та обладнання для тваринництва. Приклад застосування стрічкового методу зображений на рисунку 1.1.

Рисунок 1.1 – Приклад вимірювання стрічкою

Окрім табличних методів, існує низка формул, які дозволяють приблизно визначити живу масу ВРХ за одним-двома замірами тіла. Вони особливо корисні, коли під рукою немає відповідної таблиці або потрібно звірити результати іншими способами.

Метод Трухановського базується на обхваті грудей (О) та прямій довжині тулуба (Д) – від середини холки до кореня хвоста. Живу масу (М, кг) обчислюють за формулою:

$$М = \frac{(Д*О)*К}{100}, \quad (1.1)$$

де М – жива маса тварини, кг;

Д – пряма довжина тулуба, см;

О – обхват грудей за лопатками, см;

К – коефіцієнт, що залежить від породи та напряму продуктивності.

Коефіцієнт К враховує відмінності в будові тіла та співвідношенні м'язової і жирової тканини у тварин різних напрямів продуктивності.

Рекомендовані значення:

– К=2 для порід молочного напряму продуктивності;

– К=2.5 для порід м’ясних та м’ясно-молочних напряму продуктивності.

Отриманий результат коригують залежно від вгодованості:

– при добре вгодованій худобі до нього додають 5–10 %;

– при худій – віднімають 5–10 %.

Наявність у формулі Трухановського коефіцієнта породи (К) та поправки на вгодованість свідчить про спробу врахувати основні фактори, що впливають на співвідношення «розмір-вага». Молочні породи зазвичай мають меншу м'язову масу та жировідкладення при тому ж розмірі скелета, ніж м'ясні, що й відображає коефіцієнт К. Поправка на вгодованість додатково уточнює оцінку для конкретної тварини. Це робить метод потенційно більш адаптованим, ніж формули без таких коригувань, за умови правильного визначення типу породи та оцінки вгодованості [6].

Регресійний метод, іноді його називають формульним методом Клювера-Штрауха передбачає лише один замір обхвату грудей (X). Живу масу (Y, кг) розраховують за лінійною формулою:

$$Y = (5.3 \ * X) - c, \tag{1.2}$$

де c – константа яка змінюється в залежності від діапазону X.

Наприклад, для обхвату 181–191 см ця константа буде дорівнювати 486. Цей підхід простий у застосуванні, але не враховує довжину тіла та породні особливості.

Найпростіший спосіб – виміряти обхват живота в його найширшій частині звичайною кравецькою стрічкою, просто обгорнувши її навколо тулуба тварини. За деякими джерелами, для телят чи молодняку обхват близько 80 см дає оцінку

приблизно 30 кг, 100 см – близько 60 кг, 120 см – близько 129 кг, а 130 см – близько 160 кг живої маси, проте ці цифри дуже орієнтовні та обмежуються молодими тваринами [6].

Для дорослих корів така методика вважається найменш точною, адже обхват черева сильно змінюється залежно від наповненості шлунково-кишкового тракту, стадії тільності та індивідуальних анатомічних особливостей.

Через ці коливання показники обхвату тулуба в області живота мало корелюють із дійсною масою тіла. Набагато надійнішим вважається вимір обхвату грудей, який краще відображає розмір скелета, розвиток м’язів і вагу внутрішніх органів.

Метод обхвату черева варто використовувати лише як крайню міру для дуже приблизної оцінки, переважно у випадку телят або коли інші способи недоступні, і не покладатися на нього при прийнятті важливих рішень щодо годівлі чи лікування дорослих тварин [6].

1.2 Актуальність розробки інтелектуальної системи

Зважаючи на недоліки традиційних підходів, виникає потреба у розробці досконаліших, автоматизованих та безконтактних систем для визначення ваги худоби. Системи, що базуються на комп’ютерному зорі, мають для цього значний потенціал та низку важливих переваг.

Насамперед, вони забезпечують безконтактну оцінку ваги тварин, що істотно знижує рівень стресу для худоби та мінімізує ризик травмування як для тварин, так і для персоналу.

Крім того, сучасні візіо-системи характеризуються високим рівнем автоматизації, адже можуть працювати безперервно, майже не вимагаючи участі людини.

Це також сприяє підвищенню точності та об’єктивності вимірювань, оскільки оцінка ваги базується на даних, отриманих з зображень, що може забезпечити вищу точність порівняно з деякими традиційними методами.

Використання 3D-візіосистем, зокрема, дозволяє отримати точнішу реконструкцію тіла тварини у просторі.

Такі системи не обмежуються лише визначенням ваги, а й надавати додаткову функціональність: використовуватися для оцінки розмірів тіла, статі, породи, стану тіла (вгодованості), а також для моніторингу поведінки та виявлення аномалій, формування рекомендацій догляду [8].

Варто зазначити, що візіо-системи часто мають низькі вимоги до інфраструктури, оскільки можуть бути встановлені над загоном чи в іншому зручному місці без необхідності зведення складних конструкцій для спрямування тварин при регламентних вимірах.

Все це в сукупності також підвищує безпеку роботи, адже відсутність рухомих частин та мінімальний контакт з тваринами знижують загальні ризики [1].

1.3 Постановка задачі

Виконання кваліфікаційної роботи спрямована на створення інтелектуальної системи, призначеної для автоматизованого визначення ваги великої рогатої худоби.

Основна ідея проекту полягає у розробці веб-додатку, який дозволить користувачам завантажувати фотографії тварин та швидко отримувати оцінку їхньої живої ваги.

На основі проведеного аналізу традиційних та сучасних методів вимірювання, а також виявлених їхніх переваг та недоліків, були сформульовані конкретні вимоги до функціоналу та технічної реалізації розроблюваної системи.

1.4 Системні вимоги

Для того, щоб користувач міг взаємодіяти з розробленою ІІС оцінки ваги великої рогатої худоби через веб-сайт, йому знадобиться персональний комп'ютер або мобільний пристрій зі стабільним доступом до мережі Інтернет.

Доступ до сайту буде здійснюватися через стандартний веб-браузер, рекомендується використовувати останні версії популярних браузерів, таких як Google Chrome, Firefox, Opera, Safari.

Особливих вимог до обчислювальної потужності пристрою користувача немає, оскільки основні обчислення, пов'язані з обробкою зображень та роботою нейронної мережі, будуть відбуватися на стороні сервера.

Важливою вимогою є можливість завантаження зображень у стандартних форматах, наприклад, JPEG або PNG та достатня швидкість інтернет-з'єднання для комфортного завантаження фотографій та отримання результатів.

1.5 Вимоги до функцій програмного забезпечення

Програмний функціонал винен забезпечувати:

– завантаження зображень тварини, користувач матиме можливість завантажити одну або кілька фотографій великої рогатої худоби через зручний інтерфейс сайту;

– відображення результату оцінки ваги, система обробить завантажені фотографії, застосує інтелектуальні алгоритми та виведе на екран користувача прогнозоване значення живої ваги тварини;

– можливість отримання додаткової інформації, система також може відображати сегментоване зображення та візуальні характеристики тварини;

– можливість використовувати історію запитів користувача.

2 АНАЛІЗ ВИКОРИСТОВУВАНИХ ТЕХНОЛОГІЙ

2.1. Роль комп'ютерного зору в сучасному тваринництві

Сучасне тваринництво все більше орієнтується на принципи точного тваринництва (Precision Livestock Farming, PLF), які передбачають використання новітніх технологій для автоматичного спостереження за окремими тваринами у реальному часі.

Основна мета цього підходу – зробити виробництво більш ефективним, а також покращити здоров'я і добробут тварин завдяки переходу від загального управління стадом до індивідуальної роботи з кожною твариною. Завдяки цьому фермери можуть ухвалювати обґрунтовані рішення, спираючись на точні дані про кожну особину [9].

Однією з головних технологій, яка активно впроваджується у PLF, є комп'ютерний зір (КЗ). Він відкриває нові можливості для безконтактного збору інформації про тварин прямо на фермі. На відміну від традиційних методів, таких як ручний огляд або використання контактних пристроїв, наприклад, акселерометрів чи RFID-міток, КЗ дозволяє проводити спостереження без стресу для тварин і забезпечує цілодобовий моніторинг.

Це значно спрощує процеси, зменшує кількість необхідної ручної праці і робить оцінки об'єктивнішими [10].

Фактично, комп'ютерний зір уже став невід’ємною частиною концепції PLF. Саме завдяки йому стало можливим збирати індивідуальні дані у реальному часі та автоматично аналізувати їх для прийняття управлінських рішень. Раніше такий рівень деталізації був практично недосяжним через обмеження традиційних методів, але сьогодні КЗ дозволяє реалізувати цю ідею на практиці. Великий обсяг точних даних більше не потребує колосальних затрат праці або створення стресових ситуацій для тварин, що є величезною перевагою.

Комп'ютерний зір у моніторингу великої рогатої худоби вирішує багато важливих завдань. З його допомогою можна не тільки знаходити та визначати місцезнаходження тварин, а й виділяти їхні контури, визначати анатомічні точки та оцінювати поставу. На базі цих даних стає можливим проводити ідентифікацію тварин, автоматично вимірювати морфометричні характеристики, оцінювати їх вгодованість і навіть аналізувати поведінку для раннього виявлення проблем зі здоров'ям або добробутом

Застосування комп'ютерного зору для аналізу ВРХ спирається на низку фундаментальних технік обробки зображень та відео. Ці техніки часто утворюють послідовний конвеєр обробки, де результат одного етапу стає вхідними даними для наступного, дозволяючи вирішувати все більш складні аналітичні завдання. [9].

2.2. Аналіз інтелектуальних методів для оцінки ваги

Інтелектуальні методи, зокрема ті, що базуються на комп'ютерному зорі (КЗ), є ключовими для безконтактної оцінки параметрів ВРХ, включаючи вагу. Основні техніки КЗ включають детекцію, сегментацію та оцінку пози.

2.2.1 Детекція об’єктів

Детекція визначає наявність та місцезнаходження тварин на зображенні, зазвичай за допомогою обмежувальних рамок. Це перший крок для підрахунку, відстеження та подальшого аналізу. Для цього широко використовуються моделі глибокого навчання, такі як YOLO, Faster R-CNN та SSD, що базуються на CNN. Однак, ефективність детекції ускладнюється реальними умовами ферм: змінним освітленням, різноманітними фонами, непередбачуваними позами тварин та частими перекриттями. Для підвищення надійності застосовують механізми уваги (CBAM, SE) та трансферне навчання [11, 12].

2.2.2 Сегментація об’єктів

Сегментація забезпечує точне виділення контурів тварини на рівні пікселів, що є критичним для морфометричних вимірювань (площа, довжина, ширина) та подальшого прогнозування ваги. Розрізняють семантичну (класифікація пікселів) та інстансну (розрізнення окремих тварин) сегментацію. Для вирішення питання сегментації, використовуються такі популярні моделі, як Mask R-CNN та U-net.

Основні виклики – потреба у великих обсягах точно розмічених даних та складні умови ферми (затінення, забруднення, перекриття).

Досліджуються підходи навчання з обмеженою кількістю міток та інтеграція з 3D даними для покращення точності [13].

2.2.3 Оцінка пози об’єктів

Оцінка пози локалізує ключові анатомічні точки (суглоби, кісткові виступи) на тілі тварини для реконструкції її скелетної структури.

Використовуються спеціалізовані моделі, такі як DeepLabCut, YOLOv8-pose, HRNet, RTMPose. Основні підходи – «зверху-вниз» (спочатку детекція, потім пошук точок) та «знизу-вгору» (спочатку пошук точок, потім групування). Координати ключових точок дозволяють проводити морфометричні вимірювання, аналізувати позу та рух (виявлення кульгавості, розпізнавання поведінки), ідентифікувати тварин та вирівнювати зображення. Складність полягає у варіативності поз, перекриттях, необхідності точної анотації та проблемі доменного зсуву.

Використання 3D оцінки пози може підвищити надійність, але стикається з дефіцитом даних [14].

2.2.4 Висновок до основних технік комп’ютерного зору

Техніки КЗ (детекція, сегментація, оцінка пози) є потужним інструментарієм для безконтактного моніторингу ВРХ в рамках Точного Тваринництва (PLF). Вони уможливлюють автоматизовану ідентифікацію, фенотипування (оцінка параметрів тіла, ваги) та аналіз поведінки, сприяючи переходу до індивідуалізованого управління стадом. Це підвищує продуктивність, рентабельність та добробут тварин. Проте, широке впровадження стримується викликами, пов'язаними з надійністю алгоритмів у реальних умовах, дефіцитом даних, вартістю та складністю інтеграції технологій.

2.3. Огляд технологій та інструментів для розробки застосунку

2.3.1 Середовище розробки Visual Studio Code

Visual Studio Code (VS Code) – це безкоштовний, легкий у використанні та потужний редактор вихідного коду від Microsoft, доступний для Windows, macOS та Linux. Хоча він оптимізований для веб- та хмарної розробки, його розширювана архітектура підтримує багато мов програмування. VS Code пропонує розумне редагування коду з IntelliSense, вбудований відладчик для покрокового виконання та аналізу змінних, інтеграцію з системою контролю версій Git та вбудований термінал для виконання команд.

Його найбільша сила полягає у величезній екосистемі розширень з Marketplace, які дозволяють додати підтримку нових мов, інструментів, тем та інших сервісів, налаштовуючи середовище під конкретні потреби розробника [15].

На відміну від повноцінної IDE Microsoft Visual Studio, VS Code є передусім редактором коду, але завдяки своїй гнучкості та розширенням може

бути ефективно налаштований для широкого спектра завдань розробки програмного забезпечення на різних платформах та в різних технологічних стеках, дизайн середовища можна побачити на рисунку 2.1.

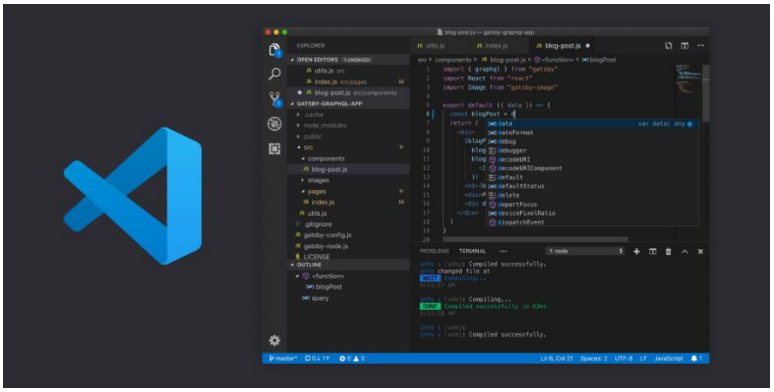

Рисунок 2.1 – Середовище розробки Visual Studio Code

2.3.2 Бібліотека Pytorch

PyTorch – це популярна відкрита бібліотека машинного навчання від Meta AI, що базується на Torch. Вона здобула визнання, особливо в дослідницькій спільноті, завдяки своїй гнучкості та ефективному використанню GPU для завдань комп'ютерного зору та НЛП.

В основі PyTorch лежать тензори – багатовимірні масиви з підтримкою GPU-прискорення, подібні до NumPy.

Однією з головних переваг є система автоматичного диференціювання torch.autograd, яка динамічно створює обчислювальні графи під час виконання програми. Це значно полегшує розробку та налагодження моделей зі змінною структурою. Для побудови нейронних мереж застосовується модуль torch.nn, для виконання оптимізації використовується модуль torch.optim, а для обробки та роботою з даними використовується torch.utils.data.

До переваг PyTorch належать його динамічні обчислювальні графи, тісна інтеграція з Python, що робить бібліотеку інтуїтивно зрозумілою, легке

перемикання між CPU та GPU, а також розгалужена екосистема спеціалізованих бібліотек (TorchVision, TorchText) та фреймворків.

Історично PyTorch відрізнявся динамічними графами від статичних у TensorFlow 1, хоча з появою TensorFlow 2 та режиму «eager execution» ця різниця зменшилась.

Традиційно TensorFlow мав переваги у промисловому розгортанні, тоді як PyTorch домінував у дослідженнях, але зараз ця межа стирається, і PyTorch активно розвиває інструменти для розгортання, як TorchServe.

Загалом, PyTorch є потужним, гнучким та зручним інструментом для глибокого навчання, що робить його доцільним вибором для вирішення завдань, поставлених у цій роботі [16].

2.3.3 Технології реалізації клієнтської частини застосунку

ReactJS – це бібліотека для створення користувацьких інтерфейсів, яка фокусується лише на шарі відображення та працює із декларативним синтаксисом JSX, що дозволяє поєднати розмітку й логіку в одному файлі.

Компоненти в React можна уявити як невеликі незалежні блоки, які приймають властивості й можуть мати власний внутрішній стан, а завдяки віртуальному DOM зміни в інтерфейсі відбуваються ефективно: спочатку оновлюється модель у пам’яті, а потім лише мінімальні відмінності застосовуються до реального деревоподібного представлення документа (рисунок 2.2).

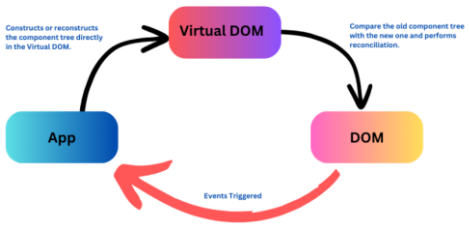

Рисунок 2.2 – Приклад роботи Virtual DOM у React

Одним із ключових здобутків React є повторне використання компонентів, розробник створює один раз універсальний елемент, наприклад кнопку чи форму, а потім застосовує його в різних частинах проєкту, що пришвидшує розробку та полегшує підтримку коду. Великий плюс цієї бібліотеки – підтримка хуків, які відкривають доступ до стану та побічних ефектів у функціональних компонентах, замінивши необхідність у класах. До того ж, завдяки численним стороннім пакетам, можна реалізувати серверний рендеринг через Next.js або нативні мобільні додатки за допомогою React Native.

Швидкий розвиток екосистеми React вимагає від розробників постійного оновлення знань, оскільки нові версії самого фреймворку та пов’язаних з ним бібліотек виходять досить часто, а офіційна документація іноді не встигає за цими змінами. React зосереджується лише на відображенні інтерфейсу, тому вибір інструментів для маршрутизації, управління станом чи здійснення HTTP-запитів покладається на розробника, що може ускладнити старт для новачків.

Крім того, синтаксис JSX може здатися незвичним тим, хто звик до чіткого поділу логіки та розмітки. Надмірне використання контексту або неефективні підходи до управління станом можуть призводити до непотрібних перерендерингів.

Насамкінець, React поєднує високий рівень продуктивності та гнучкість завдяки віртуальному DOM і хукам, проте отримає максимальну ефективність лише за умови продуманого проєктування компонентів, вибору відповідного механізму управління станом і застосування технік оптимізації, таких як мемоізація чи ліниве завантаження модулів. Це робить його відмінним рішенням для середніх і великих UI-орієнтованих проєктів, коли команда готова інвестувати час у вибір і налаштування необхідних інструментів [17, 18].

2.3.4 Технології реалізації серверної частини застосунку

Flask – це полегшений Python-фреймворк на базі WSGI, що надає базові інструменти для створення веб-застосунків: маршрутизацію URL, обробку HTTP-запитів/відповідей та сесії, покладаючись на Werkzeug та Jinja2. Його мінімалістична природа означає, що функції на кшталт доступу до баз даних чи автентифікації додаються через розширення, надаючи розробнику повний контроль над архітектурою. Хоча Flask синхронний за замовчуванням, його можна використовувати з ASGI-серверами або асинхронними розширеннями для обробки одночасних запитів. Його масштабованість залежить від архітектури та налаштувань сервера, що дозволяє витримувати значні навантаження при правильному підході [19].

Порівнюючи Flask з Django, який пропонує розробнику набір базових інструментів, таких як, Object-Relational Mapper (ORM), який представляє з себе технологію, що дозволяє розробникам працювати з даними в базі даних так, ніби це звичайні об'єкти в їхньому програмному коді, абстрагуючись від необхідності писати SQL-запити вручну, тобто фреймворк буде робити цю роботу, та адмін-панель, Flask є більш гнучким і часто обирається для API, мікросервісів та роботи з прототипами. FastAPI вирізняється вбудованою асинхронністю та автоматичною валідацією та документацією, що робить його привабливим для високонавантажених API, тоді як Flask має простішу криву навчання. На відміну від неблокуючої моделі Express.js, Flask частіше використовується в екосистемі Python, зокрема для аналітики та ML [20].

Вибір фреймворку залежить від вимог проєкту та досвіду команди. Flask пропонує свободу та легкість прототипування, але вимагає більшої архітектурної дисципліни порівняно з більш комплексними або асинхронними за замовчуванням рішеннями [21]. Порівняння Flask та FastAPI наведено в таблиці 2.1.

Таблиця 2.1 – Таблиця порівняння: Flask vs. FastAPI для розгортання ML

Характеристика	Flask	FastAPI
Продуктивність (Базова)	Добра для середніх навантажень	Дуже висока, оптимізована для I/O
Асинхронна Підтримка	Не вбудована (потребує розширень/ASGI серверів)	Нативна (ASGI)
Валідація Даних (Вбудована)	Ні (потребує зовнішніх бібліотек)	Так (Pydantic)
Автоматична Документація API	Ні (потребує розширень, напр. Flask-RESTX)	Так (Swagger UI, ReDoc)
Простота (для ML завдань)	Простий для базового API	Простий завдяки Pydantic та type hints
Екосистема (ML)	Відмінна (вся екосистема Python)	Відмінна (вся екосистема Python)
Придатність для High-Load ML	Можлива, але вимагає більше зусиль для оптимізації/асинхронності	Дуже висока

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Архітектура проекту

Процес ініціюється користувачем, котрий заходить на веб-сайт, потім, обирає розділ який йому потрібен, чи то буде сегментування, або ж отримання ключових точок, або ж ідентифікація ваги тварини, далі користувач завантажує фотографію тварини через клієнтський інтерфейс, розроблений на React. Ця клієнтська частина надсилає отримане фото на серверну частину, що працює на Flask, за допомогою HTTP-запитів. В загальному, система працює за REST-архітектурою [22].

Бекенд-сервер Flask, у свою чергу, передає це зображення двом окремим моделям машинного навчання, які були розроблені в ході кваліфікаційної роботи, а саме: моделі сегментації та моделі визначення ключових точок.

Модель сегментації обробляє зображення для виділення контуру тварини, повертаючи сегментовану маску, тоді як модель ключових точок ідентифікує тварину на фото та повертає координати специфічних анатомічних точок на тілі тварини.

Отримавши результати від обох моделей – сегментовану маску та прогнозовані ключові точки – сервер Flask передає їх до третьої розробленої моделі, яка відповідає за ідентифікацію ваги. Ця фінальна модель аналізує надані їй дані від двох інших моделей та завантажене фото тварини і розраховує прогнозовану вагу тварини.

Нарешті, обчислена вага повертається назад до клієнтської частини, яка відображає кінцевий результат користувачеві, який він потім зможе переглянути в історії передбачень. Загальна архітектура проекту зображена на рисунку 3.1.

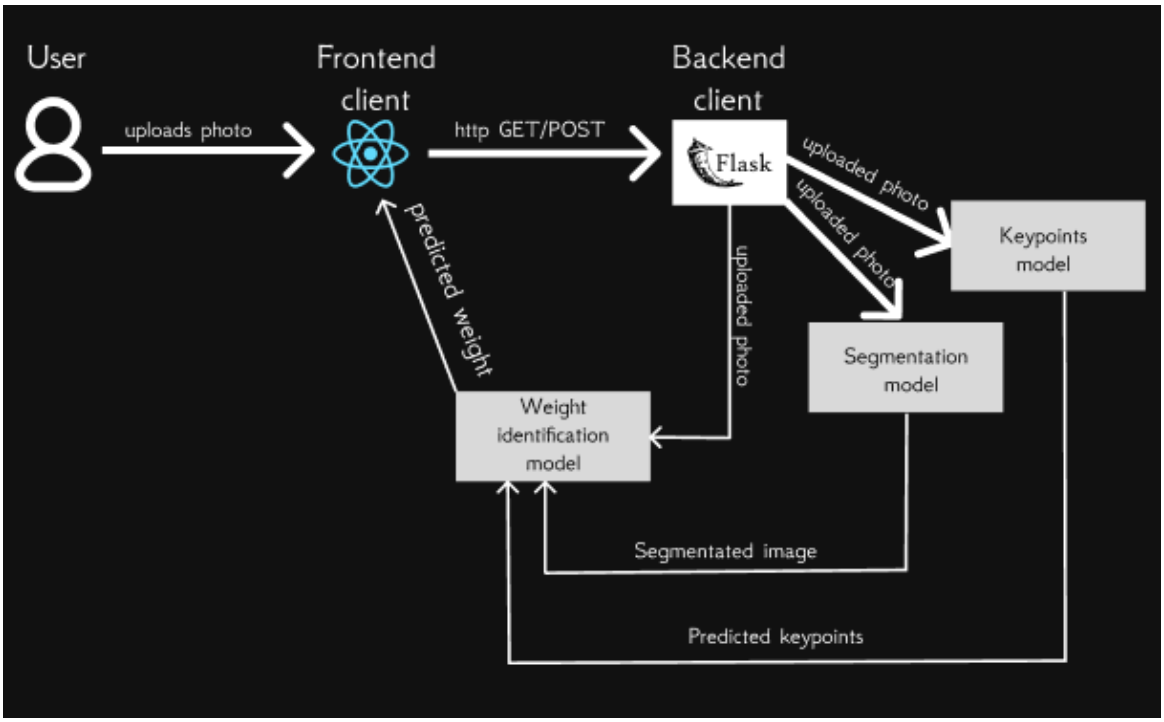

Рисунок 3.1 – Загальна архітектура проекту

3.2 Архітектура моделей інтелектуальної системи ідентифікації ваги

Для навчання та оцінки моделей використовувався набір даних «Cattle Weight Detection Model Dataset 12k» з платформи Kaggle, що містить понад 12 000 RGB-зображень ВРХ різних порід, віку та ракурсів.

В даній роботі для розробки було використано підмножину з приблизно 2600 зображень одного класу, розділену на навчальну (80%, близько 2080 зразків) та валідаційну (20%,близько 520 зразків) вибірки [23].

Датасет є релевантним завдяки наявності детальних піксельних анотацій для навчання сегментації. Вхідні зображення масштабувалися до фіксованих розмірів., 512x512 для сегментації, 224x224 для інших моделей, що прискорило навчання.

Також застосовувалася нормалізація значень пікселів для стабілізації навчання. Для підвищення стійкості та узагальнення моделей до навчальних даних застосовувалася аугментація зображень, що включала випадкові зміни

розміру, обрізку, горизонтальне віддзеркалення та зміни яскравості або контрасту; на етапі валідації застосовувалося лише детерміноване масштабування. Координати ключових точок та цільова вага для відповідних моделей були отримані з COCO та CSV файлів, дані ретельно фільтрувалися та готувалися для подачі в моделі.

Модель семантичної сегментації побудована на архітектурі UNet з енкодером ResNet-18, архітектура моделі показана на рисунку 3.2.

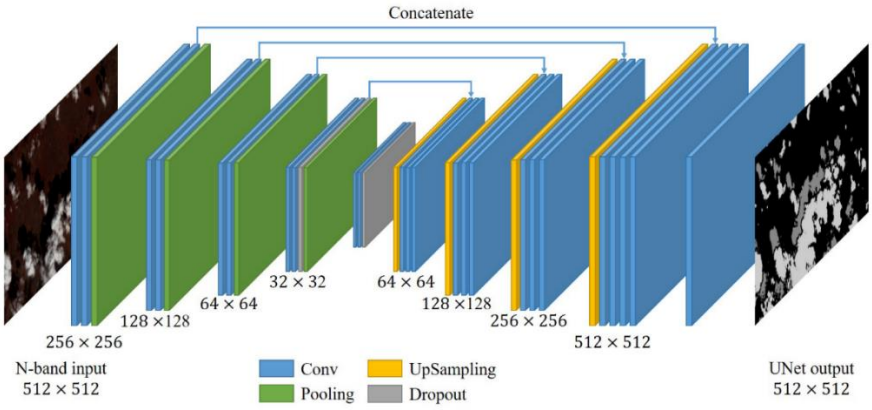

Рисунок 3.2 – Архітектура Unet моделі

Модель використовує попередньо навчені ваги ImageNet (трансферне навчання). Вона складається з енкодера, що виділяє ознаки за допомогою згорткових шарів активації ReLU [24] та шарів Max Pooling [25], які зображені на рисунку 3.3, і декодера, що відновлює просторову роздільну здатність за допомогою шарів підвищення роздільної здатності та пропускних з'єднань для точності локалізації.

$$f(x) = \max(0, x), \tag{3.1}$$

де x – це вхідне значення для функції ReLU;

Фінальний 1x1 згортковий шар видає карту сегментації для 3 класів. Модель досягла IoU близько 97%.

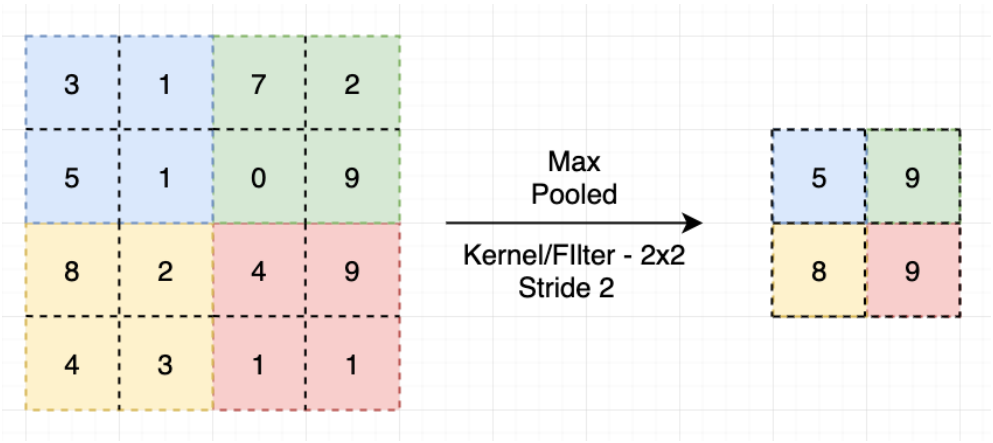

Рисунок 3.3 – Шари максимального пулінгу

Модель вилучення ключових точок використовує EfficientNetB3 як екстрактор ознак, трансферне навчання, останні шари розморожені та кастомну MLP-голову для регресії 18 координат (9 точок x, y) з лінійною активацією.

Навчання оптимізувалося задопомогою метрики RMSE,яка вимірює розбіжність між передбаченими та істинними координатами, з використанням оптимізатора Adam. RMSE обраховується за формулою:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n} + (y_i - \hat{y}_i)^2}, \tag{3.2}$$

де y_i – фактичне значення для i-го спостереження;

$\hat{y}_i$ – прогнозоване мережею значення для i-го спостереження;

n – кількість спостережень.

Як метрика для моніторингу процесу навчання використовується середня абсолютна помилка MAE, яка обраховується за формулою:

$$MAE = \frac{1}{N}\sum_{i=1}^{n}|\widehat{y_l} - y_i| \tag{3.3}$$

де n – кількість спостережень;

y_i – фактичне значення для i-ої точки даних;

$\hat{y}_i$ – передбачене значення для i-ої точки даних.

Модель оцінювалася за кастомною розробленою метрикою PCK (Percentage of Correct Keypoints), яка показує відсоток точок, передбачених з достатньою точністю відносно розміру об'єкта. Під час навчання було досягнуто PCK що прирівнюється близько 0.9894, що свідчить про те, що навчена модель функціонує адекватно.

Модель ідентифікації ваги має мультимодальну архітектуру, яка реалізована за допомогою бібліотеки PyTorch. Дві гілки на основі ResNet18 обробляють оригінальне та сегментоване зображення відповідно. Третя гілка обробляє координати ключових точок за допомогою MLP.

Ознаки з гілок конкатенуються та подаються на фінальну MLP-голову для прогнозування ваги (рисунок 3.4).

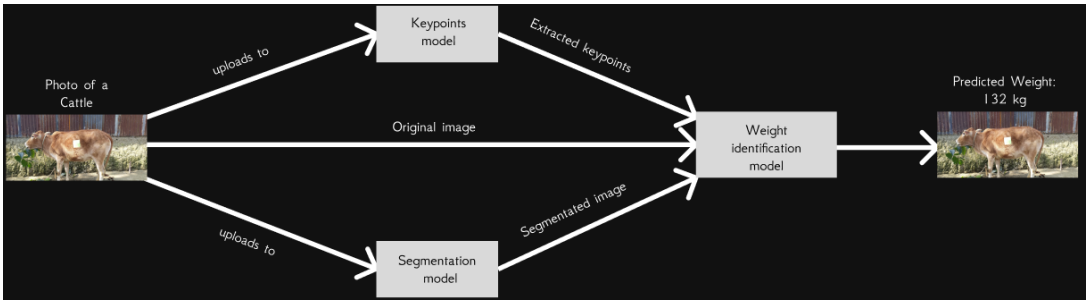

Рисунок 3.4 – Загальна мультимодальна архітектура моделі

Для навчання використовувалася функція втрат Huber Loss, яка менш чутлива до викидів, оптимізатор AdamW та розклад косинусного відпалу швидкості навчання.

Головна відмінність AdamW від Adam полягає в тому, як вони застосовують L2-регуляризацію (зменшення ваг). AdamW робить це ефективніше, відокремлюючи процес зменшення ваг від оновлення градієнтів.

Замість того, щоб додавати штраф до градієнта (як в Adam), AdamW зменшує ваги безпосередньо після їх оновлення на основі градієнта.

Це розділення в AdamW призводить до кращої генералізації моделі, більш стабільного навчання та ефективнішого контролю перенавчання, оскільки регуляризація застосовується більш передбачувано та не залежить від величини градієнтів.

Іншими словами, оптимізатор AdamW ефективніше виконує регуляризацію порівняно зі звичайним Adam. Удосконалений алгоритм AdamW базується на наступній формулі:

$$\theta_t = \theta_{t-1} - \eta * (\frac{\widehat{m_t}}{\sqrt{\widehat{v_t}+\epsilon}} + \lambda * \theta_{t-1}), \quad (3.4)$$

де η – швидкість навчання (learning rate);

ϵ – мала константа для числової стабільності;

λ – коефіцієнт weight decay;

θ – параметри моделі;

$\widehat{m_t}$ – експоненціальне згладжене середнє градієнтів;

$\widehat{v_t}$ – експоненціальне згладжене середнє квадратів градієнтів.

3.3 Архітектура клієнтської сторони застосунку

Клієнтська частина застосунку реалізована з використанням бібліотеки ReactJS, яка слугує основним інструментом для створення користувацького інтерфейсу.

Архітектура проєкту орієнтована на модульність, масштабованість і зручність у підтримці коду. Загальна структура застосунку представлена на рисунку 3.5.

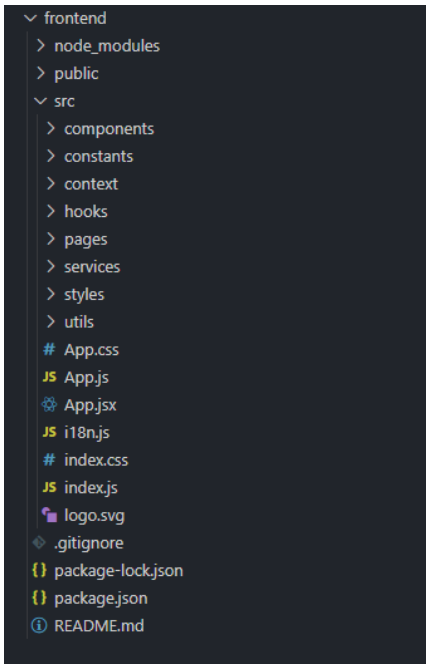

Рисунок 3.5 – Загальна структура клієнтського додатку

В загальному, розроблений додаток містить реалізацію реєстрації та логіну, яка зображена на рисунку 3.6, та має простий до розуміння користувача інтерфейс який містить три головні кнопки, історію запитів та новини проекту, реалізація зображена на рисунку 3.7.

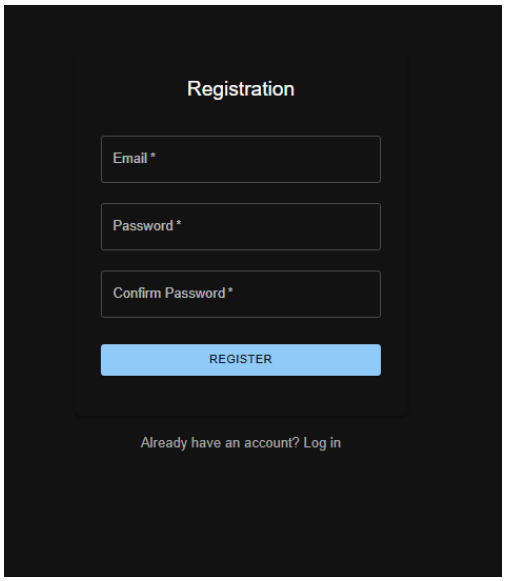

Рисунок 3.6 – Можливість реєстрації користувача

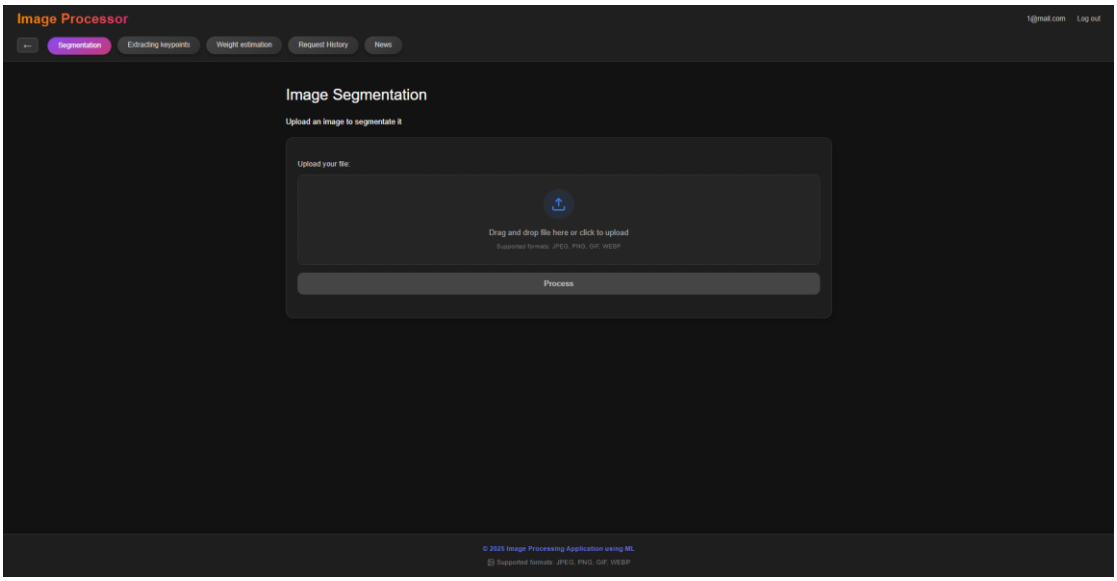

Рисунок 3.7 – Загальний вигляд розробленої клієнтської частини

В програмній реалізації використовуються функціональні компоненти, що поєднані з хуками, такими як useState, useEffect та useCallback, для ефективного управління локальним і глобальним станом, а також для контролю за життєвим циклом компонентів [26]. Приклади використання хуків зображені в лістингах 3.1-3.2.

Лістинг 3.1 – Оптимізоване завантаження файлів за допомогою useCallback

```
 const handleFileChange = useCallback((e) => {
 if (e.target.files && e.target.files[0]) {
 const file = e.target.files[0];
 if (file.type.startsWith('image/')) {
 setSelectedFile(file);
 setProcessedImage(null);
 if (showNotification) showNotification(`File
"${file.name}" chosen`, 'info');
 } else {
if (showNotification) showNotification('Please, choose an image
file', 'warning');
 }
 }
 }, [showNotification]);
```

Лістинг 3.2 – Використання хуку useEffect для отримання історії запитів користувача

```
useEffect(() => {
fetchHistory();
}, [fetchHistory]);
```

Для взаємодії з серверною частиною застосовується бібліотека Axios, яка надає зручний та гнучкий API для реалізації HTTP-запитів.

Вона дозволяє ефективно налаштовувати параметри запитів, зокрема заголовки, таймаути, та обробку відповідей і помилок. Реалізація axios-клієнту зображена в лістингу 3.3.

Лістинг 3.3 – Використання apiClient за допомогою axios

```
const apiClient = axios.create({
baseURL: `${API_BASE_URL}/api/v1`,
timeout: 60000,
});
```

Особливу роль в додатку відіграє механізм перехоплювачів, який дає змогу централізовано реалізувати додавання токенів авторизації або обробку типових помилок, таких як 401 Unauthorized. Приклад використання перехоплювачів в додатку зображений в лістингу 3.4.

Лістинг 3.4 – Використання перехоплювачів

```
apiClient.interceptors.request.use(
(config) => {
const token = localStorage.getItem('accessToken');
const noAuthUrls = ['/auth/login', '/auth/register',
'/health'];

const requestPath = config.url;
```

```
if (token && requestPath && !noAuthUrls.some(url =>
requestPath.startsWith(url))) {
config.headers['Authorization'] = `Bearer ${token}`;
console.log(`Interceptor: Added token to request for
${config.baseURL}${requestPath}`);
} else {
 console.log(`Interceptor: No token added for
${config.baseURL}${requestPath}`);
}
return config;
},
(error) => {
console.error("Interceptor request error:", error);
return Promise.reject(error);
});
```

Система маршрутизації реалізована за допомогою бібліотеки React Router DOM, яка є стандартом для організації клієнтської навігації у React-застосунках. Вона дозволяє створювати повноцінні односторінкові застосунки (SPA), де перемикання між віртуальними сторінками відбувається без повного перезавантаження [27].

Роутинг керується за допомогою ключових компонентів, таких як BrowserRouter, Routes і Route, а також через використання хуків useNavigate, useParams та useLocation, що забезпечують гнучке управління навігацією. Приклад використання роутингу в додатку зображений в лістингу 3.5.

Лістинг 3.5 – Використання роутингу для переходу на сторінку запитів користувача

```
<Route path="/history" element={
     <ProtectedRoute>
          <HistoryPage />
     </ProtectedRoute>
} />
```

Для організації глобального стану застосовуються два основні підходи. У простіших сценаріях використовується вбудований механізм React Context API, який дозволяє уникати передачі пропсів на глибоких рівнях дерева компонентів.

Контекст доцільно використовувати для таких даних, як тема оформлення, статус аутентифікації або загальні результати, що мають бути доступні у багатьох частинах інтерфейсу [28].

В проєкті було розроблено AuthContext, який відповідає саме за аутентифікацію.

Компонент AuthContext слугує централізованим механізмом для управління всім, що стосується аутентифікації користувача у застосунку, використовуючи вбудований інструмент React під назвою Context API.

По суті, він створює спеціальний «контейнер» даних, який зберігає актуальну інформацію про стан аутентифікації: чи увійшов користувач в систему, дані самого користувача, його унікальний токен доступу, а також відстежує процес завантаження та можливі помилки, що виникають під час логіну чи реєстрації.

При першому завантаженні застосунку AuthContext автоматично перевіряє, чи є збережений токен у локальному сховищі браузера. Якщо токен знайдено, компонент намагається перевірити його валідність, запитуючи дані поточного користувача з сервера. Залежно від успішності цієї перевірки, він оновлює стан, позначаючи користувача як аутентифікованого або скидаючи недійсний токен.

Окрім цього, AuthContext надає зручні функції для взаємодії з системою аутентифікації, логін для входу користувача, лог-аут для виходу та регістрація для створення нового акаунту. Ці функції інкапсулюють логіку взаємодії з відповідними API-ендпоінтами на сервері, обробляють відповіді, оновлюють внутрішній стан компонента та зберігають або видаляють токен з локального сховища.

Найважливіше те, що спеціальний механізм AuthContext працює як центральний вузол, який роздає всім частинам застосунку актуальну інформацію, про те, хто зараз увійшов, чи відбувається завантаження, а також надає готові «кнопки» для входу та виходу. Завдяки цьому будь-який елемент

програми, якому це потрібно, легко отримує доступ до цих даних та функцій з одного надійного місця.

Це досягається без необхідності передавання даних через багато рівнів компонентів. Для зручності доступу до цих даних та функцій зазвичай створюється спеціальний хук useAuth, який інші компоненти можуть легко використовувати. Компонент також дбає про те, щоб відображувати не основний інтерфейс програми, а стан завантаження під час початкової перевірки токена. Загальний приклад процесу реєстрації користувача наведено в лістингу 3.6.

Лістинг 3.6 – Використання хука useCallback для реєстрації користувача в контексті AuthContext

```
const register = useCallback(async (credentials) => {
setIsLoading(true);
setAuthError(null);
try {
const data = await registerUserAPI(credentials);
console.log("AuthProvider:     Registration     successful:",
data.message);
setAuthError(null);
setIsLoading(false);
return true;
} catch (error) {
console.error("AuthProvider: Registration failed:", error);
 setAuthError(error?.error || error?.message || 'Registration
failed');
 setIsLoading(false);
 return false;
}
}, []);
```

Для оптимізації окремих аспектів розробки були використані додаткові бібліотеки. Зокрема, для стилізації інтерфейсу застосовано Material UI [29], що забезпечує готові компоненти, підтримку дизайн-системи, вбудовані засоби доступності, а також зручну колекцію іконок, які активно використовувалися під час оформлення застосунку. Один із прикладів використання Material UI наведено в лістингу 3.7.

Лістинг 3.7 – Використання прогрес-бару з інструменту Material UI для візуалізації логіну.

```
import CircularProgress from '@mui/material/CircularProgress';

{isLoading && (
 <CircularProgress
 size={24}
 sx={{
 position: 'absolute',
 top: '50%',
 left: '50%',
 marginTop: '-12px',
 marginLeft: '-12px',
 }}/>)}
```

Для інформування користувача про різноманітні події в додатку, такі як успішне виконання дії або виникнення помилки, була реалізована кастомна система сповіщень.

Візуальною частиною цієї системи є компонент NotificationSystem. Цей компонент відповідає безпосередньо за відображення повідомлень на екрані.

Він приймає текст сповіщення як властивість notification і динамічно з'являється у верхньому правому куті сторінки лише тоді, коли є активне повідомлення для показу; в іншому випадку він залишається невидимим.

Його зовнішній вигляд, включаючи фіксоване позиціонування, кольорову схему, тінь та анімацію появи й легкого коливання, визначається за допомогою вбудованих стилів безпосередньо в коді, що робить його самодостатнім візуальним елементом цієї розробленої системи нотифікації подій в додатку, результат роботи цієї системи зображена на рисунку 3.8.

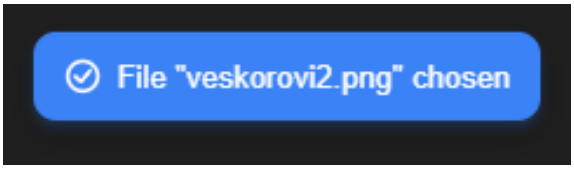

Рисунок 3.8 – Робота NotificationSystem

Також для поліпшення інформованості користувачей було розроблено механізм експорту історії запитів користувача за бажанням: в історії запитів створено кнопку «Export in JSON», що зображено на рисунку 3.9.

Рисунок 3.9 – Кнопка експортування історії запитів користувача

Натискаючи на кнопкуінтерфейсу, користувач може безпосередньо отримати свої запити, збережені на комп’ютері. Загальний код реалізації функції експорту даних наведено в лістингу 3.8.

Лістинг 3.8 – Реалізація експорту історії користувача

```
const handleExportJson = useCallback(() => {
if (!historyData || historyData.length === 0) {
console.warn("No history data to export.");

return;
}

try {
const jsonString = JSON.stringify(historyData, null, 2);

const blob = new Blob([jsonString], { type: "application/json"
});

const url = URL.createObjectURL(blob);

const link = document.createElement('a');
link.href = url;
const timestamp = new Date().toISOString().replace(/[:.]/g, '-
');
link.download = `image_processing_history_${timestamp}.json`;

document.body.appendChild(link);
link.click();
```

```
document.body.removeChild(link);
URL.revokeObjectURL(url);

console.log("History exported successfully.");

} catch (err) {
console.error("Failed to export history to JSON:", err);
}

}, [historyData]);
```

3.4 Архітектура серверної сторони застосунку

Серверна частина застосунку розроблена з використанням мікрофреймворку Flask на мові Python, що забезпечує легкість та гнучкість реалізації. Як показано в лістингу 3.9, додаток ініціалізується разом з необхідними розширеннями, такими як SQLAlchemy для роботи з базою даних, Bcrypt для хешування паролів, Migrate для міграцій та JWTManager для автентифікації. Використання CORS дозволяє взаємодію з клієнтськими застосунками з інших доменів [30, 31].

Архітектура побудована за принципами REST API, надаючи клієнтській частині інтерфейс для взаємодії через HTTP-запити.

Лістинг 3.9 – Ініціалізація Flask та розширень

```
# Initialize Flask app
app = Flask(__name__)
CORS(app)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.config['JWT_SECRET_KEY'] =
os.environ.get('JWT_SECRET_KEY', 'your-super-secret-key-
replace-me')
db = SQLAlchemy(app)
migrate = Migrate(app, db)
bcrypt = Bcrypt(app)
jwt = JWTManager(app)
```

Як система керування базами даних використовується SQLite, а взаємодія з нею здійснюється через ORM Flask-SQLAlchemy.

Схема бази даних включає моделі для зберігання інформації про користувачів та історію обробки зображень, схема бази даних наведена на рисунку 3.10.

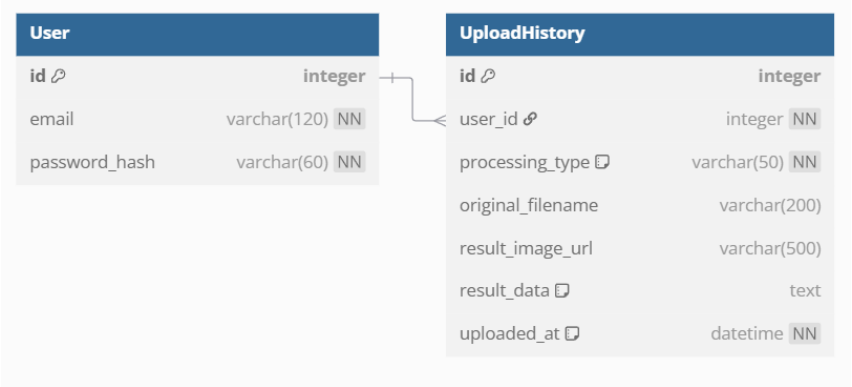

Рисунок 3.10 – Загальна схема бази даних

Ця модель пов'язана з користувачем і зберігає тип обробки, імена файлів, URL результатів та самі результати, статистику сегментації, координати точок або прогнозований вагу, у форматі JSON.

Для реалізації історії користувача була розроблена таблиця UploadHistory, яка зберігає всі запити користувача які він зробив в додатку (лістинг 3.10).

Лістинг 3.10 – Модель бази даних для історії

```
class UploadHistory(db.Model):
id = db.Column(db.Integer, primary_key=True)
user_id = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=False)
processing_type = db.Column(db.String(50), nullable=False)
original_filename = db.Column(db.String(200))
result_image_url = db.Column(db.String(500))
result_data = db.Column(db.Text, nullable=True)
uploaded_at = db.Column(db.DateTime, nullable=False,
default=datetime.utcnow)
user = db.relationship('User',backref=db.backref('uploads',
lazy=True))
```

Автентифікація користувачів реалізована за допомогою JSON Web Tokens (JWT), керованих бібліотекою Flask-JWT-Extended. Реєстрація та логін користувачів здійснюються через ендпоінти /api/v1/auth/register (POST) та /api/v1/auth/login (POST) відповідно, при цьому паролі безпечно хешуються за допомогою Flask-Bcrypt [32].

Більшість функціональних ендпоінтів, таких як обробка зображень чи отримання історії, захищені та вимагають валідний JWT токен для доступу, що реалізується за допомогою декоратора @jwt_required() [33].

Сервер надає набір REST API ендпоінтів для різних функцій. Окрім маршрутів аутентифікації, є ендпоінт /api/v1/users/me (GET) для отримання інформації про поточного користувача та /api/v1/history (GET) для отримання історії його обробок.

Основна функціональність пов'язана з обробкою зображень: /api/v1/segmentation (POST) виконує сегментацію за допомогою моделі PyTorch UNet, /api/v1/keypoints (POST) визначає ключові точки за допомогою моделі TensorFlow, Keras, а /api/v1/weight (POST) інтегрує результати попередніх кроків для прогнозування ваги за допомогою мультимодальної моделі PyTorch.

Таблиця 3.1 – Таблиця ендпоінтів

Ендпоінт	Метод	Потребує JWT
/api/v1/auth/register	POST	Ні
/api/v1/auth/login	POST	Ні
/api/v1/users/me	GET	Так
/api/v1/segmentation	POST	Так
/api/v1/keypoints	POST	Так
/api/v1/weight	POST	Так
/api/v1/history	GET	Так

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Система та авторизація в ній

При першому заході на сайт користувач отримує сторінку авторизації для входу, або ж якщо в нього немає акаунту він може перейти на сторінку реєстрації та з легкістю його створити (рисунок 4.1).

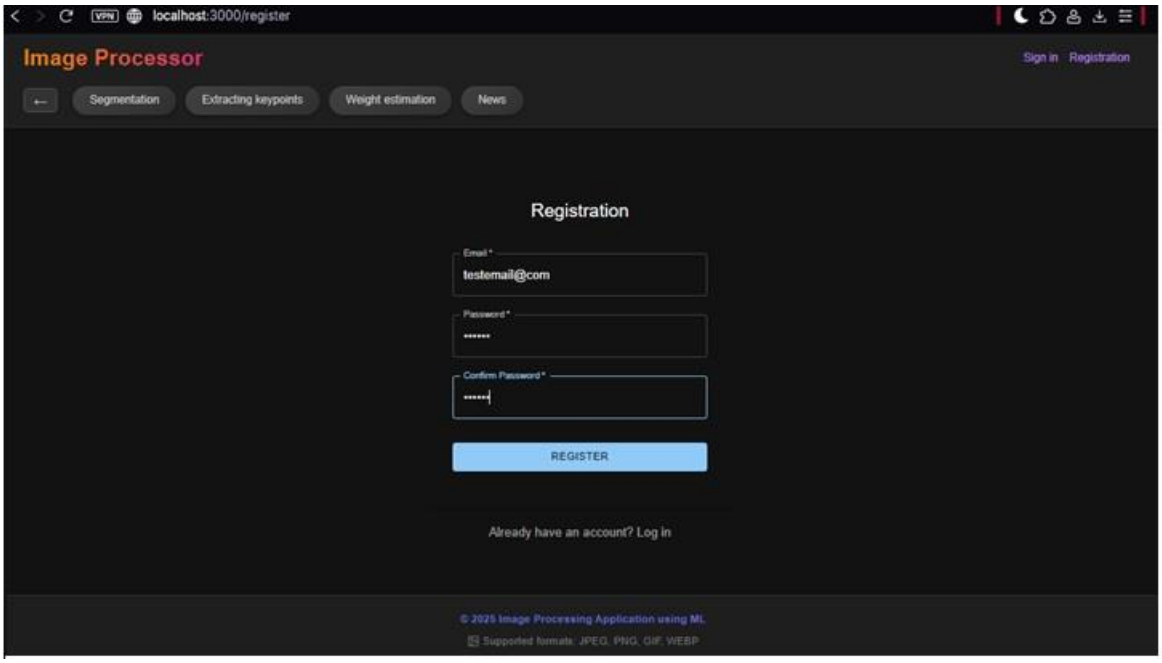

Рисунок 4.1 – Реєстрація нового користувача в системі ідентифікації

Механізм валідації електронної пошти на стороні клієнта є важливим елементом для забезпечення коректності даних та покращення користувацького досвіду.

Коли користувач взаємодіє з формою введення, наприклад, під час реєстрації або входу в систему, і вводить свою електронну адресу, система не просто пасивно приймає ці дані. Система перевіряє, чи відповідає введений текст стандартному формату електронної адреси [34].

Якщо введені дані не проходять цю перевірку, наприклад, користувач забув ввести @, ввів неприпустимі символи, або залишив поле порожнім, – система негайно реагує (рисунок 4.2).

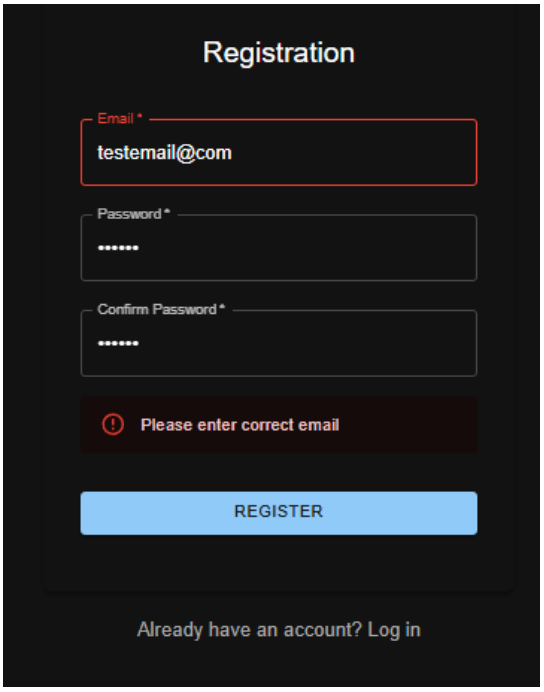

Рисунок 4.2 – Валідація помилок некоректного введення емейлу

Аналогічно до перевірки формату електронної пошти, для поля введення пароля також реалізований схожий механізм клієнтської валідації, спрямований на забезпечення базових вимог безпеки та інформування користувача

Однією з таких типових перевірок є мінімальна довжина пароля. Система аналізує кількість введених символів і порівнює її з встановленим мінімальним значенням. Якщо користувач ввів пароль, що коротший за цю вимогу наприклад, лише 5 символів, валідація не пройде.

У такому випадку, так само як і з полем email, система надасть користувачеві негайний зворотний зв'язок (рисунок 4.3).

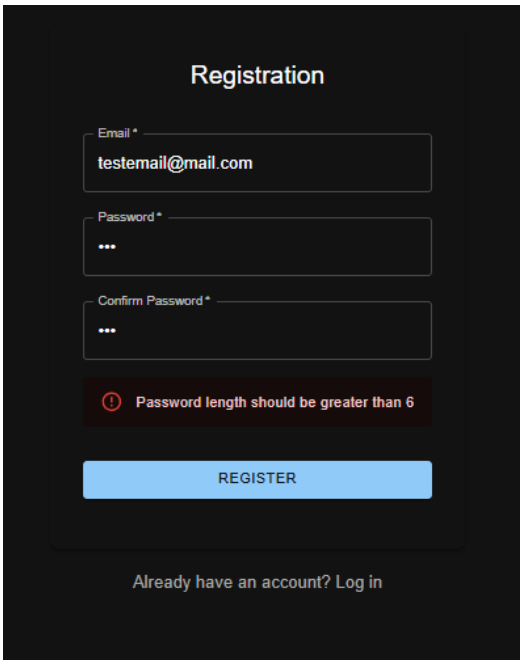

Рисунок 4.3 – Валідація помилок некоректного введення паролю

4.2 Функціонал системи

Після успішної авторизації, користувач, має доступ до повного функціоналу системи, користувача зустрічає приємне меню з темною темою та головне меню, яке містить наступні пункти (рисунок 4.4):

– сегментація зображень тварини;

– вилучення ключових точок тварини;

– ідентифікація ваги тварини;

– історія запитів;

– новини проєкту.

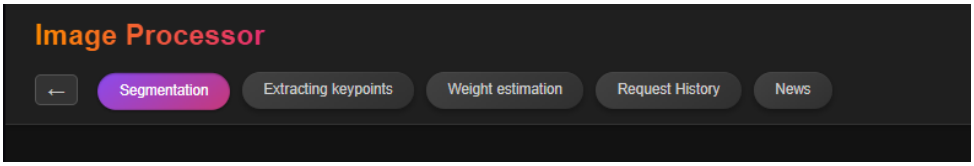

Рисунок 4.4 – Головне меню системи

Далі, користувач обирає що він хоче зробити, сегменувати зображення, отримати ключові точки, або ж відразу отримати вагу тварини, для прикладу оберемо сегментацію, користувачу доступне меню, яке є достатньо зрозумілим для кожного користувача, яке містить поле для завантаження файлу, користувач може перетягнути фото або ж обрати фото на персональному комп'ютері (рисунок 4.5).

Після того, як користувач обрав фото, він натискає кнопку «Process», яка відповідає за обробку фото, після чого користувач отримує бажаний результат (рисунок 4.6), який потім зберігається в його історії запитів.

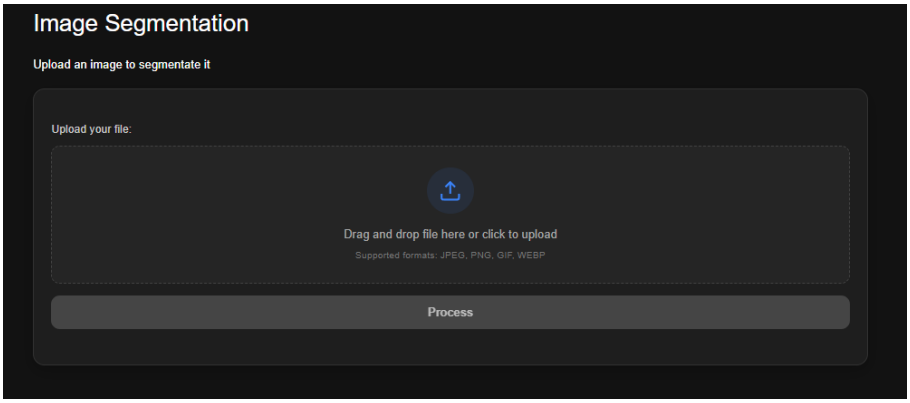

Рисунок 4.5 – Меню сегментації зображень

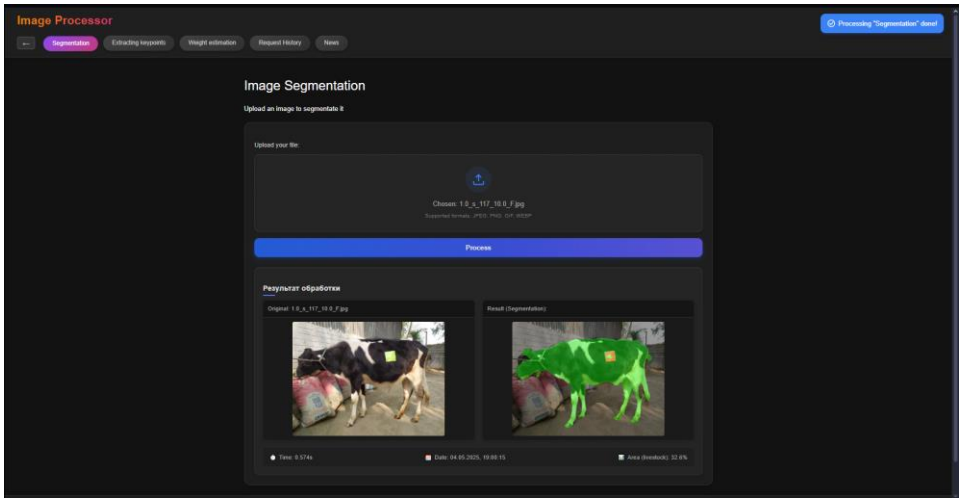

Рисунок 4.6 – Результат сегментації зображень

За таким же принципом, користувач має змогу отримати ключові точки завантаживши зображення (рисунок 4.7), та отримати вагу тварини (рисунок 4.8).

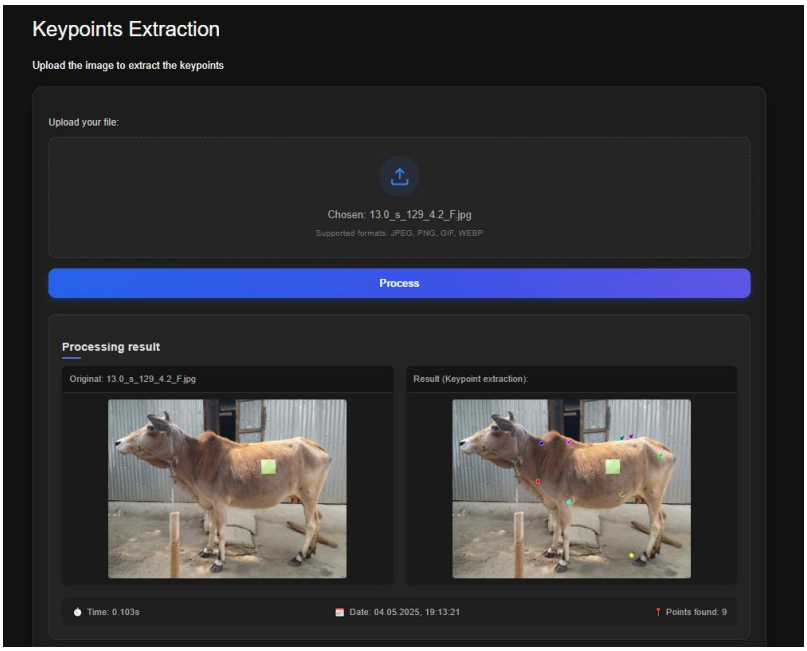

Рисунок 4.7 – Отримання ключових точок

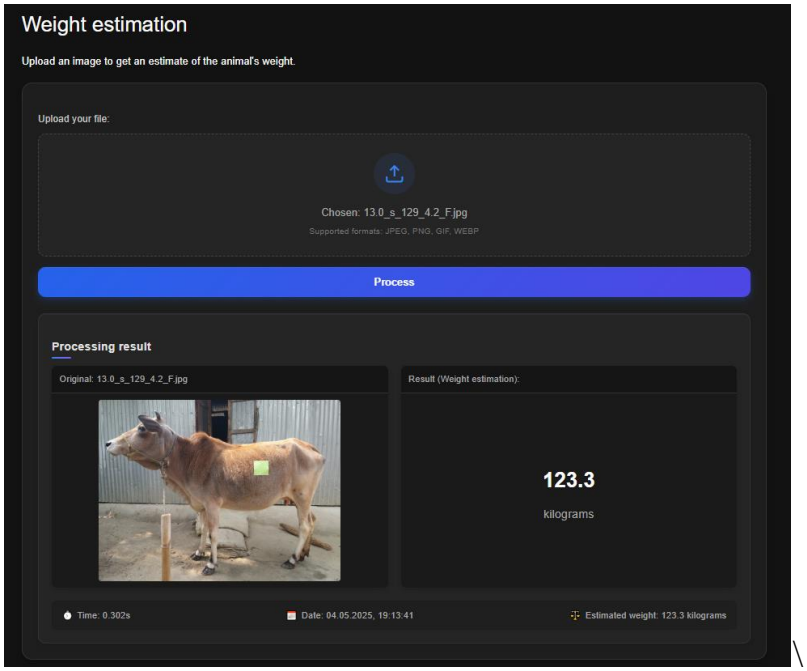

Рисунок 4.8 – Ідентифікація ваги тварини

Крім того, користувач має можливість переглядати та експортувати історію своїх запитів до моделей, що суттєво покращує взаємодію з системою та підвищує зручність її використання (рисунок 4.9).

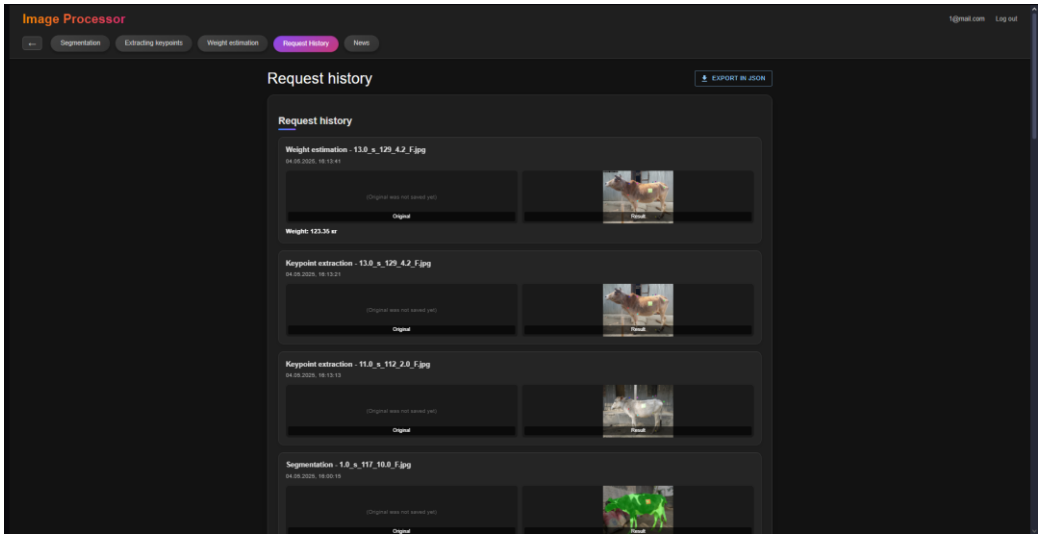

Рисунок 4.9 – Історія запитів користувача

Більш того, натискаючи кнопку «Export in JSON», користувач отримує файл зі всіма його запитами (рисунок 4.10)

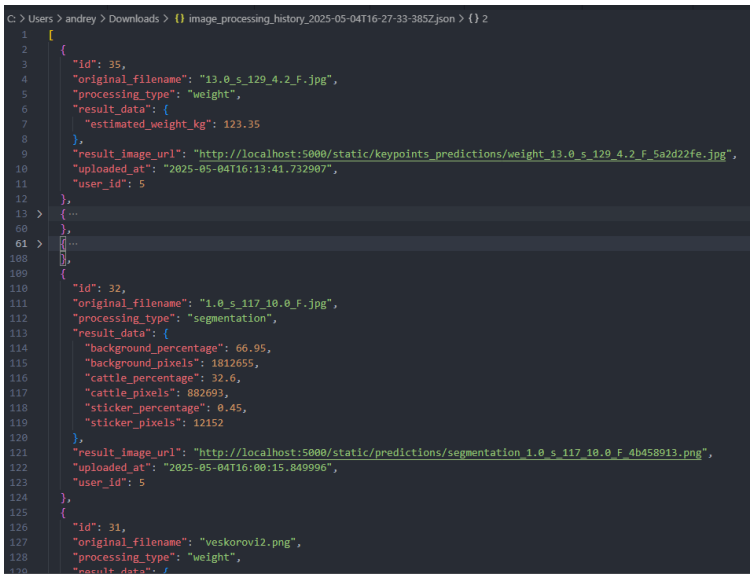

Рисунок 4.10 – Завантажений файл-історія користувача

Для тримання кожного користувача в курсі всіх подій було розроблене базове меню новин, яке дозволяє перейти до нього та прочитати останні новини або ж чому моделі можуть не працювати (рисунок 4.11).

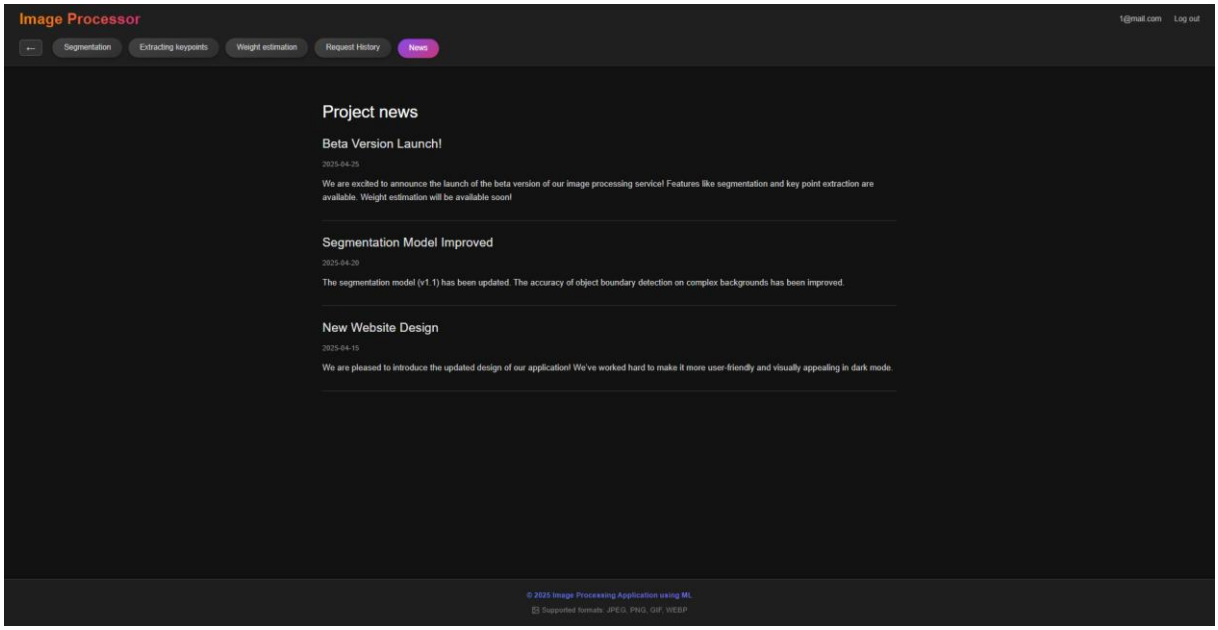

Рисунок 4.11 – Стрічка новин

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було досягнуто поставлену мету, розроблено інтелектуальну систему для автоматизованої ідентифікації ваги великої рогатої худоби на основі аналізу зображень. Було проведено аналіз предметної області, який виявив суттєві недоліки традиційних методів визначення ваги ВРХ, такі як низька точність візуальної оцінки, стрес для тварин під час ручних вимірювань, залежність від типу та калібрування інструментів, таких як стрічки та формули, а також значні трудовитрати.

Розроблена інтелектуальна система демонструє потенціал для значного покращення процесу визначення ваги ВРХ порівняно з традиційними методами. Вона забезпечує безконтактну оцінку, що знижує стрес для тварин та ризики для персоналу, автоматизує процес, зменшуючи потребу в ручній праці, та має потенціал для підвищення точності вимірювань. Це, в свою чергу, може сприяти покращенню управління стадом, оптимізації годівлі, своєчасному ветеринарному втручанню та підвищенню загальної економічної ефективності фермерських господарств.

Реалізований веб-інтерфейс надає зручний доступ до функціоналу системи, включаючи історію запитів та можливість експорту даних.

Таким чином, у кваліфікаційній роботі було представлено спроектований програмний комплекс, що вирішує актуальну задачу автоматизації оцінки ваги ВРХ за допомогою методів комп'ютерного зору та глибокого навчання. Подальший розвиток системи включає розширення набору даних для навчання моделей, покращення їх стійкості до різних умов зйомки та порід тварин, інтеграцію додаткових функцій аналізу стану худоби, розробка системи індивідуального харчування худоби, а також інтеграція платіжних систем для монетизації та розвитку моделей.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Weighing in on Weight Tapes. *Dairy Herd*. URL: https://www.dairyherd.com/news/weighing-weight-tapes (дата звернення: 28.04.2025).

2. Visual weight estimation and the risk of underdosing dairy cattle. *ResearchGate*. URL: https://www.researchgate.net/publication/275717473_Visual_weight_estimation_and_the_risk_of_underdosing_dairy_cattle (дата звернення: 28.04.2025).

3. Comparison and reliability of techniques to estimate live cattle body. *Taylor & Francis Online*. URL: https://www.tandfonline.com/doi/full/10.1080/09712119.2017.1302876 (дата звернення: 28.04.2025).

4. A weight tape worth $1,000. *Hoard's Dairyman*. URL: https://hoards.com/article-25961-a-weight-tape-worth-$1000.html (дата звернення: 28.04.2025).

5. Coburn Weigh Tape. *PBS Animal Health*. URL: https://www.pbsanimalhealth.com/coburn-weigh-tape/p/11273/ (дата звернення: 28.04.2025).

6. Як дізнатися вагу корови без ваг. *Ava Market*. URL: https://avamarket.com.ua/porady-expertiv/vrh/kak-uznat-ves-korovy-bez-vesov (дата звернення: 28.04.2025).

7. Zerodis 2.5m Cattle Body Weight Tape Measure, Red, PVC, 250cm/98.4in, Push-Button Lock, 2.04% Accuracy, Ideal for Farmers. *Amazon.com*. URL: https://www.amazon.com/Zerodis-Measure-Push-Button-Accuracy Farmers/dp/B08SQH1LRQ (дата звернення: 28.04.2025).

8. Черкасов А.Д., Ілюнін О.О. Інтелектуальна модель ідентифікації великої рогатої худоби, XXIX МІЖНАРОДНИЙ МОЛОДІЖНИЙ ФОРУМ «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ». Україна. 2025. С. 169-171.

9. Computer vision algorithms to help decision-making in cattle production. *PMC (PubMed Central)*. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC11700597/

(дата звернення: 28.04.2025).

10. Computer Vision-Based Measurement Techniques for Livestock Body Dimension and Weight: A Review. *MDPI*. URL: https://www.mdpi.com/2077-0472/14/2/306 (дата звернення: 28.04.2025).

11. Role of computer vision and deep learning algorithms in livestock behavioural recognition: A state-of-the-art- review. *Edelweiss Applied Science and Technology*. URL: https://learning-gate.com/index.php/2576-8484/article/view/3396 (дата звернення: 28.04.2025).

12. A review of three-dimensional computer vision used in precision livestock farming for cattle growth management. *ResearchGate*. URL: https://www.researchgate.net/publication/368691074_A_review_of_three- dimensional_computer_vision_used_in_precision_livestock_farming_for_cattle_growth_management (дата звернення: 28.04.2025).

13. Intelligent weight prediction of cows based on semantic segmentation and back propagation neural network. *Frontiers in Artificial Intelligence*. URL: https://www.frontiersin.org/journals/artificial- intelligence/articles/10.3389/frai.2024.1299169/full (дата звернення: 28.04.2025).

14. Animal Pose Estimation: A Closer Look at the State-of-the-Art, Existing Gaps and Opportunities. *OpenReview*. URL: https://openreview.net/pdf?id=OqzTgrcTWx (дата звернення: 28.04.2025).

15. Visual Studio Code. URL: https://code.visualstudio.com (дата звернення: 05.05.2025).

16. PyTorch. URL: https://pytorch.org (дата звернення: 05.05.2025).

17. ReactJS Introduction. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/reactjs-introduction/ (дата звернення: 05.05.2025).

18. ReactJS | Virtual DOM. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/reactjs-virtual-dom/ (дата звернення: 05.05.2025).

19. Flask - Overview. *Tutorialspoint*. URL: https://www.tutorialspoint.com/flask/flask_overview.htm (дата звернення: 05.05.2025).

20. Django vs Flask: Which Python Framework to Choose? *Cloudways Blog*. URL: https://www.cloudways.com/blog/django-or-flask/ (дата звернення: 05.05.2025).

21. FastAPI vs Flask: A Detailed Comparison. *Turing.com*. URL: https://www.turing.com/kb/fastapi-vs-flask-a-detailed-comparison (дата звернення: 05.05.2025).

22. REST APIs. *IBM*. URL: https://www.ibm.com/think/topics/rest-apis (дата звернення: 05.05.2025).

23. Cattle Weight Detection Model Dataset 12K. *Kaggle*. URL: https://www.kaggle.com/datasets/sadhliroomyprime/cattle-weight-detection-model-dataset-12k (дата звернення: 05.05.2025).

24. ReLU Activation Function in Deep Learning. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/relu-activation-function-in-deep-learning/ (дата звернення: 05.05.2025).

25. CNN | Introduction to Pooling Layer. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/ (дата звернення: 05.05.2025).

26. Built-in React Hooks. *React*. URL: https://react.dev/reference/react/hooks (дата звернення: 05.05.2025).

27. React Router. URL: https://reactrouter.com (дата звернення: 05.05.2025).

28. Context. *Legacy React Docs*. URL: https://legacy.reactjs.org/docs/context.html (дата звернення: 05.05.2025).

29. Material UI. URL: https://mui.com/material-ui/ (дата звернення: 05.05.2025).

30. Flask-JWT-Extended Documentation. URL: https://flask-jwt-extended.readthedocs.io/en/stable/ (дата звернення: 05.05.2025).

31. SQLAlchemy. URL: https://www.sqlalchemy.org (дата звернення: 05.05.2025).

32. Password Hashing with Bcrypt in Flask. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/password-hashing-with-bcrypt-in-flask/ (дата звернення: 05.05.2025).

33. Flask-JWT Documentation. URL: https://pythonhosted.org/Flask-JWT/ (дата звернення: 05.05.2025).

34. How to Validate Emails in React. *Mailtrap Blog*. URL: https://mailtrap.io/blog/validate-emails-in-react/ (дата звернення: 05.05.2025).