КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ:

«Програмний модуль виявлення хвороб шкіри обличчя за допомогою згорткової нейронної мережі»

Галузь знань 12 «Інформаційні технології»
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Комп'ютерні науки»
Освітній рівень: бакалавр

Виконав:
студент 4 курсу групи КН-42
Комарніцький І.В.

Керівник:
Сорока Петро Миколайович,
кандидат фізико-математичних наук, доцент

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри інтелектуальних технологій
Протокол №_13__від_05.06.2023 р.
Зав. кафедри _____ доц. Іларіонов О.Є.

Київ – 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ:
Зав. кафедри інтелектуальних технологій

_____
(звання, прізвище та ініціали)

_____
(підпис)

«_____»___2023 р.

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Комарніцькому Іллі Вікторовичу

(прізвище, ім'я, по батькові)

1. Тема роботи « Програмний модуль виявлення хвороб шкіри обличчя за допомогою згорткової нейронної мережі» затверджена наказом по університету від « 11 » листопада 2022 р. протокол №4

2. Термін здачі студентом закінченого роботи 02.06.2023

3. Вихідні дані до роботи розробити сервіс для виявлення висипань на шкірі обличчя

4. Зміст пояснювальної записки (перелік питань, що їх належить розробити)
   1) аналіз технологій з розпізнання захворювань шкіри
   2) розробка архітетури модулю розпізнання захворювань шкіри
   3) технічні особливості реалізіації програмного модуля

5. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій
   1.Обйєкт та предмет дослідження , мета роботи (1 слайд)
   2.Актуальність розробки застосунку та аналіз наявних систем (3 слайди)
   3.Проектні рішення з реалізації веб-застосунку(5 сладів)
   4.Програмна реалізація веб-застосунку(2 слайди)
   5.Тестування веб-застосунку (6 сладів)
   6.Висновок (1 слайд)

6. Консультанти з випускної кваліфікаційної роботи із зазначенням її розділів, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання

Керівник_____ / Сорока П. М. /
(підпис) (ініціали та прізвище)

Завдання прийняв до виконання_____ / Комарніцький І.В /
(підпис) (ініціали та прізвище)

КАЛЕНДАРНИЙ ПЛАН

Пор.№	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів випускної кваліфікаційної роботи	Примітка
1	Опрацювання літератури	15.02 – 27.02	
2	Робота над розділом 1 аналіз технології розпізнавання захворювань шкіри за допомогою нейро мережі	27.02 – 08.03	
3	Робота над розділом 2. розробка архітектури модулю розпізнання захворювань шкіри	08.03 – 08.04	
4	Робота над розділом 3. технологічні особливості реалізації програмного модуля	08.04 – 13.05	
5	Робота над оформленням пояснювальної записк	13.05 – 25.05	
6	Робота над презентацією	25.05 – 29.05	
7			
8			
9			
10			

Студент _____ис) / Комарніцький І.В. /
(ініціали та прізвище)

Керівник випускної кваліфікаційної роботи_____ / Сорока П .М. /
(підпис) (ініціали та прізвище)

Анотація

Комарніцький Ілля Вікторович виконав кваліфікаційну роботу на тему «Програмний модуль виявлення хвороб шкіри обличчя за допомогою згорткової нейронної мережі» за спеціальністю 122 – «Комп'ютерні науки».

У кваліфікаційній роботі розроблено сервіс для розпізнання проблем із висипаннями на шкірі обличчя. Сервіс реалізовано у вигляді web-додатку, завдяки якому користувачі зможуть отримати результат виялення захворювань , які допоможуть їм в подальшому лікуванні.

Ключові слова: акне, класифікація захворювань шкіри обличчя, згорткова нейронна мережа, функція активації, web-додаток.

Summary

The degree project: «Software module for detection of facial skin diseases using a convolutional neural network» has been completed by Komarnitskyi Ilya in specialty 122 – «Computer Science».

In the qualification work, a service was developed for recognizing problems with rashes on the skin of the face. The service is implemented in the form of a web application, thanks to which users will be able to receive the results of the detection of diseases, which will help them in further treatment.

Keywords: acne, classification of facial skin diseases, convolutional neural network, activation function, web application.

ЗМІСТ

Вступ

Акне є одним з найпоширеніших захворювань як в підлітків, так і в дорослих, що впливає не тільки на зовнішній вигляд людини, а і на її самопочуття. Люди, які хворіють захворюваннями шкіри мають послаблений імунітет через постійний запальний процес. Захворювання шкіри призводять до почуття незадоволеності життям та гіршому ментальному здоров'ю, що є однією з найголовніших причин низької самооцінки у підлітків. Також акне свідчить про деякі більш серйозні проблеми зі здоров'ям, найчастіше це ознака поганої роботи: кишечника, печінки, підшлункової залози, жовчного міхура та майже всіх органів ШКТ. Також захворювання шкіри можуть бути спричинені нездоровим способом життя: вживання алкоголю, куріння, малорухливий спосіб життя, недостатня кількість випитої чистої води за день тощо. Висипання на шкірі також можуть бути спричинені поганим доглядом за нею: відсутність мила для обличчя, тоніка, нехтування кремом для зволоження або обезжирювання шкіри.

Для того, щоб реалізувати web-сервіс для сканування наявності акне на обличчі було використано нейромережу по пошуку захворювань шкіри, та на основі даних, які були отримані після сканування та введення користувачем опису.

Даний сервіс може бути використаний як в косметологічних цілях, так і в медицинських та для покращення стану шкіри.

Метою кваліфікаційної роботи є розробка web-сервісу завдяки якому люди могли б швидко розпізнати хворобу шкіри обличчя та отримати результати виявлення захворювання, які допомогли б в подальшому лікуванні.

Об'єктом дослідження кваліфікаційної роботи є люди, які страждають від висипань на обличчі, та профілактика такого роду висипань.

Предметом дослідження є алгоритми та методи виявлення захворювань шкіри обличчя.

Завдання дослідження. Основним завданням дослідження для теми розробки сервісу по розпізнанню акне є:

- Розробка алгоритмів та моделей навчання для виявлення захворювань шкіри обличчя.
- Порівняння різних підходів виявлення акне.
- Визначення оптимальних моделей, які матимуть найвищу точність.
- Розробка web-додатку, щоб користувач міг просто отримати всі потрібні йому дані.
- Тестування та валідація web-сервісу з використанням реальних зображень захворювань шкіри обличчя.
- Дослідження можливості використання системи в різних галузях, такі як косметологія, охорона здоров’я, фармацевтична галузь.

РОЗДІЛ 1 АНАЛІЗ ТЕХНОЛОГІЇ З РОЗПІЗНАВАННЯ ЗАХВОРЮВАНЬ ШКІРИ ЗА ДОПОМОГОЮ НЕЙРО МЕРЕЖІ

1.1 Аналітичний огляд предметної області створення веб-сервісів детекторів акне

У багатьох випадках люди, які страждають захворюваннями шкіри не мають достатньо коштів, щоб звернутись до косметолога та виконувати поради, які той надасть. Тому вони часто починають займатись самолікуванням не володіючи повною інформацією щодо вирішення проблеми, а це може бути небезпечно або навіть нашкодити їм та погіршити їх стан. Отже виникає потреба у швидкому та самостійному виявленню акне та покращанню стану шкіри.

Серед невеликої кількості визначень веб-сервісів, виявлених у літературі, найбільш офіційним визначенням є «Веб-служба, веб-сервіс (англ. web service) - програмна система, що ідентифікується URI, і публічні інтерфейси та прив’язки якої визначені та описані мовою XML».

Термін «веб-служба» застосовується до багатьох різних систем, але в основному він стосується клієнтів та серверів, що взаємодіють за допомогою повідомлень протоколу SOAP. В обох випадках припускається, що існує також опис доступних операцій у форматі WSDL. Хоча наявність цього опису не є вимогою SOAP, а радше передумовою для автоматичного генерування коду на платформах Java та .NET на стороні клієнта.

Технології веб-сервісів, що базуються на можливостях Інтернету, кардинально покращують взаємодію людей та інформаційних систем одне з одним. Вони утворені з цілого ряду стандартних протоколів взаємодії, засобів опису моделей даних та інтерфейсів, а також допоміжних мережевих служб, що забезпечують доступність бізнес-функцій організацій авторизованим користувачам через Інтернет з будь-якого підключеного до нього пристрою. Зокрема, веб-сервіси дозволяють:

- описати їх у вигляді сервісу і забезпечити до них доступ користувачів ззовні;
- знайти такий сервіс сторонам, зацікавленим у його використанні;
- скористатися даним сервісом після його виявлення;
- забезпечити інтерпретований результат взаємодії.

Таким чином веб-сервіси забезпечують побудовану на відкритих стандартах інформаційну інфраструктуру, за допомогою якої організації можуть:

- інтегрувати внутрішні бізнес-процеси один з одним;
- динамічно пов’язувати і синхронізувати власні бізнес-процеси з бізнес-процесами своїх ділових партнерів;
- пропонувати свої бізнес-процеси в якості сервісів, якими можуть скористатися інші організації на певних умовах.

Веб-сервіси служать основою для чергового, ще більш масштабного, перетворення, в результаті якого Інтернет вже набуває рис універсального ділового середовища, в якій безперешкодно протікають всілякі ланцюжки процесів публікації, виявлення і споживання різних бізнес-сервісів.

На чому базуються веб-сервіси? По суті Веб-сервіси є новий вид веб-додатків для створення рівня бізнес-логіки і зв'язку різнорідних додатків на основі використання загальних стандартів. Завдяки веб-сервісам функції будь-якої прикладної програми стають доступними через Інтернет.

Всі веб-сервіси реалізуються на загальних принципах:

- створювач конкретного веб-сервісу визначає формат запитів до нього і формат відповіді на дані запити;
- з будь-якого комп’ютера в Інтернет можна зробити запит до даного веб-сервісу;
- веб-сервіс виконує задану послідовність дій і відправляє назад результат.

1.2 Аналітичний огляд методів виявлення захворювань шкіри

1.2.1 Метод виявлення захворювань шкіри

При виявленні захворювань шкіри за допомогою цифрового зображення можна виділити 3 рівня захворювань шкіри, які треба буде лікувати за різними сценаріями та при використанні інших підходів та засобів: 1-фото в якому багато засвітів та фільтрів (рис. 1.1а); 2-відео (рис. 1.1б); 3-фото (рис. 1.1в), зроблене нещодавно та без фільтрів. Фотографії, які були зроблені нещодавно, сприяють підвищенню рівня точності виялення акне.

Для того, щоб підтримати та полегшити складні дослідження щодо виявлення акне, зібрано новий набір даних під назвою AcneSCU. Порівняно з попередніми наборами даних для виявлення акне, AcneSCU має більш високу роздільну здатність і нормовані зображення, дрібнозернисті категорії акне і більш точні анотації.

Зображення AcneSCU з високою роздільною здатністю спочатку обрізає зображення, але часткові ураження акне на контурі внесуть неточну навчальну інформацію. Щоб обійти це питання, запропоновано простий, але ефективний метод попередньої обробки даних, який називається masked crop, маскуючий всі часткові ураження.

Нова структура під назвою SADH пропонується для вирішення проблеми плоского градієнта впевненості класифікації. Навчання репрезентації для класифікації та локалізації здійснюється двома різними шарами згортки, що створює більш крутий градієнт впевненості класифікації .

Для прогнозування впевненості в локалізації пропонується гілка прогнозування NWD разом із втратою SBCE. Було продемонстровано, що класифікаційні оцінки, виправлені NWD, можуть покращити не тільки кореляцію між оцінками класифікації та IoUs пропозицій, але й загальну ефективність виявлення.

а) б)

в)

Рисунок 1.1 – Різні способи виявлення захворювань шкіри: а) фото в якому багато засвітів та фільтрів; б) відео; в) фото, зроблене нещодавно та без фільтрів

1.2.1.1. Виявлення обличчя за допомогою фото з фільтром

Якщо використовувати фото, які оброблені або мають засвіт, це буде сильно знижувати точність роботи нейромережі, так як не буде відображати реальну картину стану шкіри. Тому такий метод тут не підходить. Було використано двошаровий алгоритм виявлення, який базується на технології глибокого навчання для виявлення акне. Результати показали, що він

перевершував одношаровий алгоритм за середньою швидкістю запам’ятовування.

1.2.1.2 Виявлення обличчя за допомогою зйомки в реальному часі.

Це може підійти тільки в тому випадку, якщо людина має змогу знімати на якісну камеру, без засвітів, бо інакше виникнуть проблеми з обробленням даного відео.

1.2.1 Виявлення обличчя з нещодавно зробленою необробленою фотографією

Без засвітів буде найкращим варіантом, так як найбільш точно передає реальний стан шкіри. Проаналізуваши таку фотографію, можна буде зробити найбільш точний результат виялення акне. При використанні детектора Coarse-to-Fine Network (CFN) [1] було виявлено, що середня точність становила 89%, а середня тривалість виконання – 0,3 с.

1.2.2 Методи надання рекомендацій по догляду

Надання рекомендацій передбачає розбиття нашого процесу на етапи. Це розбиття на етапи показано на рис. 1.2. Обрізаємо кожне зображення на кілька підзображень. Однак велика кількість уражених місць на одне зображення ускладнює забезпечення того, щоб кожне ураження було обрізано повністю. Збереження або ігнорування місця обрізаних часткових уражень призведе до великої кількості неточних зображень і неповноцінної продуктивності. У роботі пропонується простий, але ефективний спосіб, маскуючи всі часткові ураження. Кожне підзображення обрізано розміром 1024×1024. Щоб забезпечити всю область зображення для участі в навчанні, загальна кількість ⌈W1024⌉×⌈H1024⌉ підзображень обрізається, де W і H позначає ширину і висоту вихідного зображення відповідно. Кожне зображення спочатку обрізається по горизонтальному напрямку з рівними перекриттями, а потім таким же чином

обрізається по вертикальному напрямку. Оскільки існує перекриття між сусідніми підзображеннями, мало місць про ураження опущено. Потім застосовуються алгоритми AcneSCU, RPN [2].

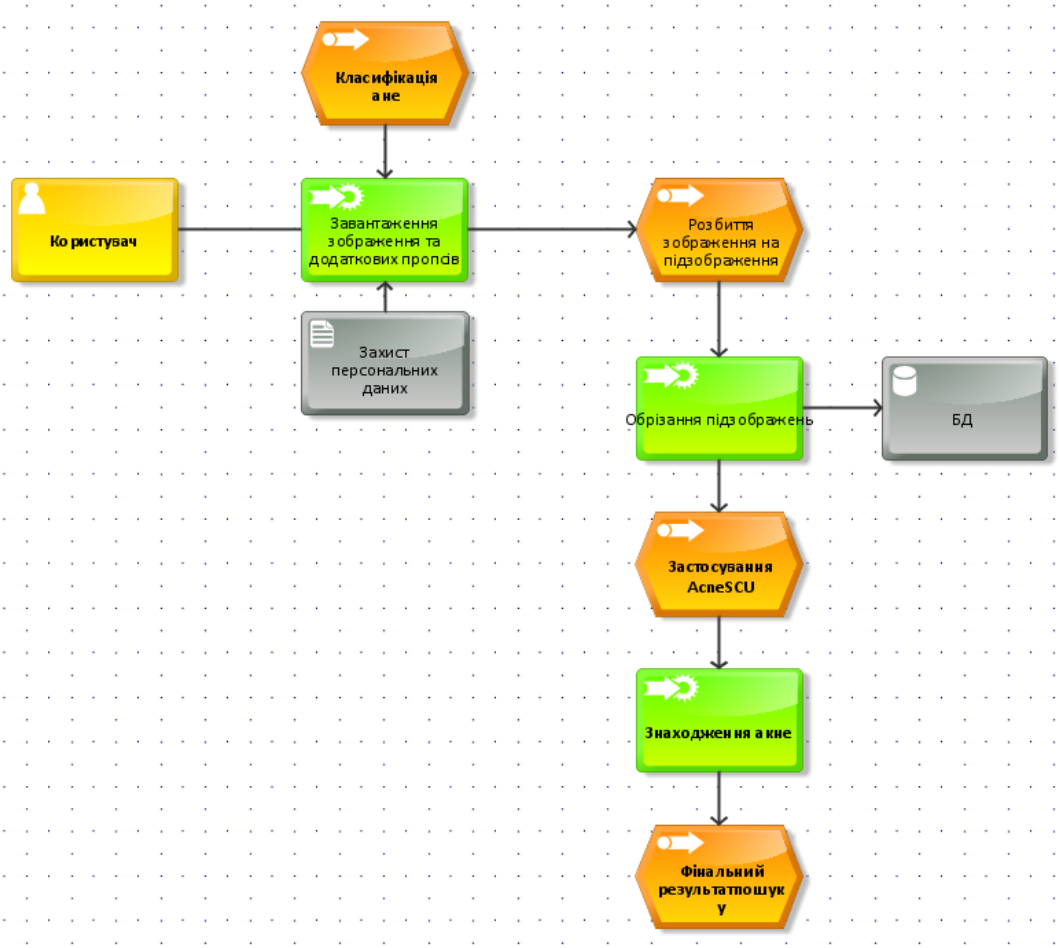


Рисунок 1.2 – Діаграма методів рекомендацій по догляду

Згорткова нейронна мережа (ЗНМ) складається з багатьох об’єднаних шарів. Згорткові шари виконують зважені згортки між своїми вхідними параметрами та їх вагами, які можна вивчати. З цього випливає, що вони знаходять локальні закономірності у вхідних даних. Шари об’єднання – це шари, які не підлягають навчанню, які зменшують розмірність свого введення шляхом локального відображення невеликого квадрата на вході в одне число. Класифікатор зазвичай складається з одного або кількох повністю пов’язаних шарів і функції softmax. Порівняємо основні типи ЗНМ.[3]

RPN широко використовуються при виявленні об’єктів і тонко розроблені для різних завдань. RPN, широко прийнятий у двоступеневих методах виявлення, по суті, є свого роду повністю згортковою мережею (FCN) повністю для генерації пропозицій на кожній позиції карти функцій. RPN складається з двох відгалужень для завдань класифікації та локалізації відповідно. LAW запропонував для прогнозування верхнього лівого кута та нижнього правого кута об’єкта. Ще більше розширив Cornernet, передбачивши центральну точку об’єкта. MEA [11] запропонував передбачити відповідний градусний бал між прогнозованою рамкою розміру та основною істиною, а також LOU запропонував універсальну схему дослідження відносин для дослідження відносин різного рівня сутностей, таких як об’єкти, суперпікселі та пікселі. Маска R-CNN [4] розширила швидше R-CNN, додавши головку сегментації екземпляра. WUA провели ретельне порівняння між повністю з’єднаною R-CNN і виявили цікаве явище: він більше підходить для класифікаційної задачі, в той час як LAW більше підходить для задачі локалізації.

AcneNet: це глибинна нейронна мережа, розроблена для розпізнавання акне на обличчі. Модель навчалась на наборі даних, що містив 10 000 зображень облич з акне та без акне. AcneNet використовує архітектуру ResNet та досягає точності більше 90%.

MobileNetv2 [5] — це легка згорткова мережа для мобільних додатків. Він містить кілька розділених по глибині згорток, які можна розглядати як згорткові шари, де тензор тривимірних ваг розкладається на один двовимірний тензор і одновимірний тензор, що вимагає набагато менше пам’яті. Цей тип шарів часто використовується в легких згорткових мережах. Що стосується інших мереж, MobileNetv2 має менше нелінійності. Автори дають інтерпретацію, чому це повинно дати кращі результати в їхній мережі, і стверджують, що та сама мережа працювала гірше, коли вони намагалися додати більше лінійності. Крім того, ця мережа має інвертовані пропуски підключень. Це означає, що приховані шари, з’єднані за допомогою пропускних з’єднань, мають низькі розміри, що зменшує кількість операцій, які виконує мережа.

DeepAcne [6]: це інша глибинна нейронна мережа, яка розпізнає акне на зображеннях облич. Модель навчалась на наборі даних з більш ніж 10 000 зображень та використовує технології передобробки зображень, такі як обрізання, розмиття та підсвічування, для покращення результатів.

GoogleNet [7] складається з 22 шарів CNN. Він має лише 4 мільйони параметрів, що є дуже низьким числом порівняно з 60 мільйонами параметрів AlexNet. Основною особливістю GoogleNet є використання початкових шарів. Вони виготовлені з різних згорткових шарів різного розміру, виходи яких об’єднані на кінці модуля. Ідея полягає в тому, що фільтри різного розміру можуть виявляти різні шаблони вхідних даних.[9]

DenseNet201 є великою та високопродуктивною згортковою мережею. Його назва походить від того, що кожен шар пов’язаний з усіма попередніми. Ця архітектура допомагає градієнтним потокам і заохочує повторне використання функцій. DenseNet201 є конкурентоспроможним з іншими найсучаснішими мережами ImageNet, маючи менші параметри, ніж AlexNet.

1.3. Порівняльний аналіз вже існуючих детекторів акне

SkinVision[16] – це мобільний додаток, який дозволяє користувачам аналі зувати зображення шкіри з метою виявлення пігментних плям, зневоднення шкіри та акне. SkinVision використовує штучний інтелект, щоб забезпечити користувачів точними результатами діагностики.

Рисунок 1.3. Приклад конкурентів SkinVision

MDacne [17] - це додаток для мобільних пристроїв, який аналізує зображення обличчя з метою виявлення акне. Додаток також пропонує індивідуальні поради щодо догляду за шкірою на основі типу шкіри та рівня важкості акне.

Рисунок 1.4. Приклад конкурентів MDacne

SkinVisionary [15] - це онлайн-сервіс, який використовує штучний інтелект для діагностики дерматологічних захворювань, включаючи акне. Користувачі можуть завантажити зображення своєї шкіри та отримати детальний звіт про стан своєї шкіри.

Рисунок 1.5. Приклад конкурентів Designer Skin

Cognitec (https://www.cognitec.com/) надає клієнтам FRS, який можна масштабувати та налаштовувати, завдяки своїй відкритій архітектурі системи через «FaceVacs». Cognitec найкраще підходить для організацій, які шукають рішення безпеки та контролю натовпу на основі FRS. Він має значну присутність на ринку розпізнавання облич і може надавати ефективні масштабні рішення з мінімальними витратами на налаштування.

Рисунок 1.6. Приклад конкурентів Cognitec

DeepVision AI (https://deepvisionai.in/) надає рішення FRS для маркетингу та планування, а також для компаній, які хочуть використовувати перевірку облич з метою безпеки. DeepVision найкраще підходить для забудовників нерухомості, фірм роздрібної торгівлі та маркетингових агентств. Функція на основі штучного інтелекту тут може бути застосована для безпеки та мобільності пішоходів, виявлення інцидентів і розпізнавання транспортних засобів, серед іншого, для забезпечення автоматизованого аналізу відео. Він ідеально підходить для середніх і великих підприємств, яким потрібні масштабовані рішення.

Рисунок 1.7. Приклад конкурентів deepvision

Face++ (https://www.faceplusplus.com/) чудово підходить для порівняння облич зі збереженими зображеннями. Деталізація функцій, які він пропонує, і його гнучкі моделі ціноутворення роблять його природним вибором для компаній, які все ще намагаються з’ясувати, як вони хочуть інтегруватися з FRS.

Рисунок 1.8. Приклад конкурентів Face++

1.4 Постановка задачі

При розробці програмного продукту була поставлена задача створити web-сервіс завдяки якому люди могли б швидко визначити діагноз та придбати всі необхідні препарати, які допоможуть їм побороти проблеми з висипаннями на обличчі, з результатами отриманими після сканування продовжити лікування. Для розробки front-end частини нашого додатку було обрано мову програмування Java script, а саме фреймворк React [19] з використанням бібліотеки Next.js [20], а для роботи зі стилями використовувся препроцесор SCSS [24] на основі CSS з використанням методології BOM. У якості бази даних було викристано postgres SQL, так як ця реаліаційна база даних тут чудово підходить. Для розробки back-end було використано мову програмування next.JS. У якості нейронної мережі було використано згорткову нейромежу RPN та алгоритм AcneSCU .[9]

1.5 Перелік основних вимог

Функціональні вимоги:

- можливість почати розпізнання захворювання на обличчі;
- можливість розпізнавати захворювання на обличчі;
- можливість закінчити розпізнання захворювання на обличчі;

- можливість змінювати інтерфейс веб-додатку;

Нефункціональні вимоги:

- зручний у використанні інтерфейс веб-сайту;
- зручний каталог товарів;
- мова інтерфейсу – українська, англійська.

1.6 Аналіз зацікавлених сторін

1. Внутрішні зацікавлені сторони
− косметологи;
− біологи;
− організації по вивченню акне;
− фармацевти;
− дерматологи.
2. Зовнішні зацікавлені сторони
− міністерство охорони здоров’я України;
− фармацевтичні організації;
− конкурентні організації;
− звичайні люди. які використовують сервіс для своїх цілей.

Протагоністами можуть бути розробники системи, її користувачі та косметологи, які сканують проблеми з обличчям. Антагоністами можуть бути конкуренти, які вже мають свої рішення для пошуку акне, або правові органи, які можуть обмежувати використання нейромережі.

Висновок першого розділу

Отже, перший розділ кваліфікаційної роботи включає в себе опис області застосування детектора акне, аналітичний огляд методів захворювань шкіри, виявлення зацікавлених сторін проекту. В ньому розглянуто наявні рішення (існуючі сервіси з аналізу обличчя на акне), представлено короткі теоретичні відомості про можливі важкості акне, що будуть використовуватись у програмі,

а також детально описана постановка задачі (тема, мета, об’єкт, предмет та завдання досліджень, функціональні та нефункціональні вимоги до програмного продукту).

РОЗДІЛ 2. РОЗРОБКА АРХІТЕКТУРИ МОДУЛЮ РОЗПІЗНАННЯ ЗАХВОРЮВАНЬ ШКІРИ

2.1. Функціональний аналіз

Сучасні технології розпізнання об’єктів, які мають за мету розпізнання різних об’єктів (у нашому випадку захворювань шкіри) значно покращились. Уже існують методи для реалізації завдання розпізнання об’єктів з високою точністю і невисокими апаратними затратами. Ці методи глибинного навчання комп’ютерного зору (КЗ), можна умовно поділити на 2 етапи: 1- побудова моделі; 2- аналіз зображень за допомогою моделі.

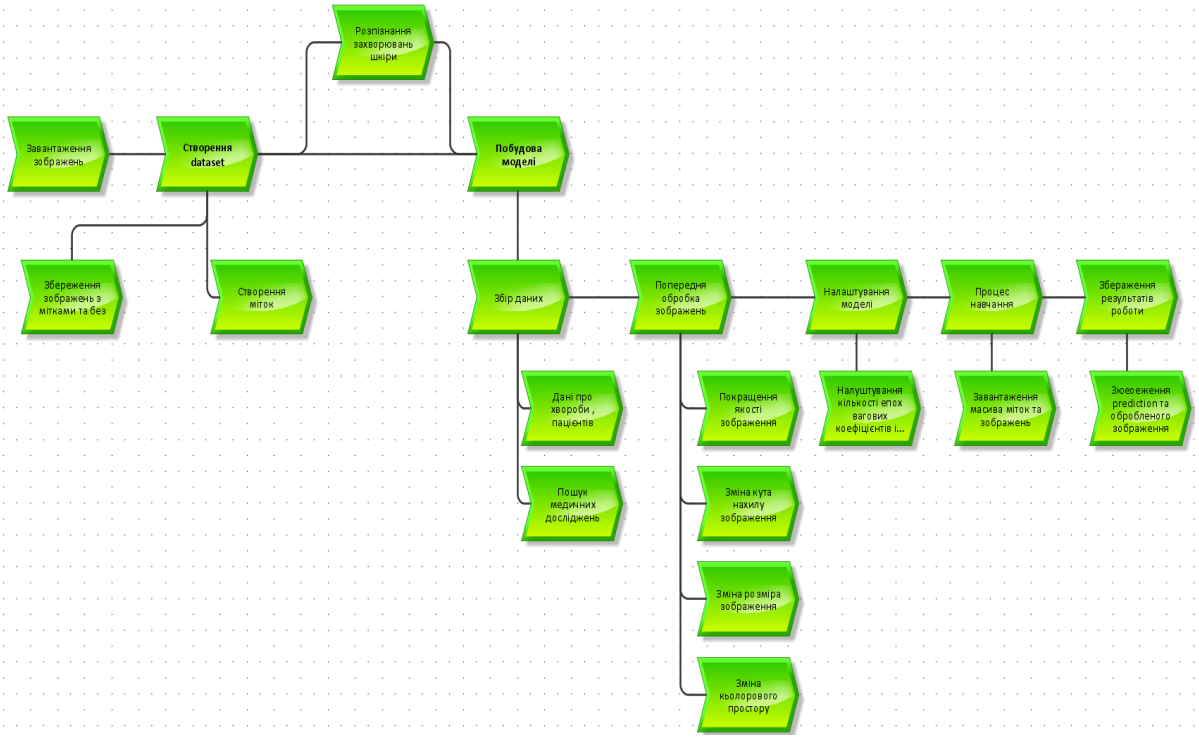


Рисунок 2.1 − Дерево функцій модуля acne detection

Опис кожного етапу.

Побудова моделі включає в себе такі підетапи:

1. Збір даних:

- зібрати відповідні дані про хворобу та пацієнта, такі як фотографії пошкодження шкіри, симптоми та інші важливі медичні дані;

- пошук вже готових наборів даних (kaggle [23], tensorflow [25], Hugging Face, Yollo);[22]

- пошук медичних дослідів вже зі збірником зображень та їх описом.

2. Попередня обробка зображення:

- корекція розміру зображення (зменшення або збільшення, розбиття на підзображення, задавати фіксований розмір для оптимальної роботи;

- корекція кута нахилу зображення (поворот зображення на *n*-ний кут, зсув);

- покращення якості зображення (зменшення або збільшення шуму, підвищення контрасту, накладення відповідних фільтрів або масок);

- зміна кольорового простору (RGB – складається з поєднання 3-х кольорів, а це в свою чергу може не завжди буде доцільним, тому його можна перетворити в інший кольоровий простір, наприклад HSV, CMY).

3. Налаштування моделі

- налаштування моделі (кількість епох, встановлення коефіцієнтів, бажана точність).

4. Процес навчання моделі

- завантажити масив зображень та міток в модель (з розподілом на категорії (Training set – для тренування, Valid set – для валідації, Testing Set – для тестування отриманих результатів).

5. Збереження результату роботи моделі

- збереження розультатів (prediction, та оброблене зображення)

При підготовці dataset можна виділити наступні під етапи:

- завантеження зображення на Yollo;

- створення annotated для зображень та встановлення міток для виявлення захворювань шкіри;

-збереження масивів з обробленими та необробленими зображеннями, у форматі Yollo-4.

2.2. Архітектура системи у вигляді IDEF0

Забезпечує:

− мову для побудування моделі рішень та операцій в організації, процес.

− визначення, задокументування, повідомлення про основну діяльність системи.

− дає більш чітке розуміння як система пов'язана між собою.

− показує області діяльності, які потрібно вдосконалити.

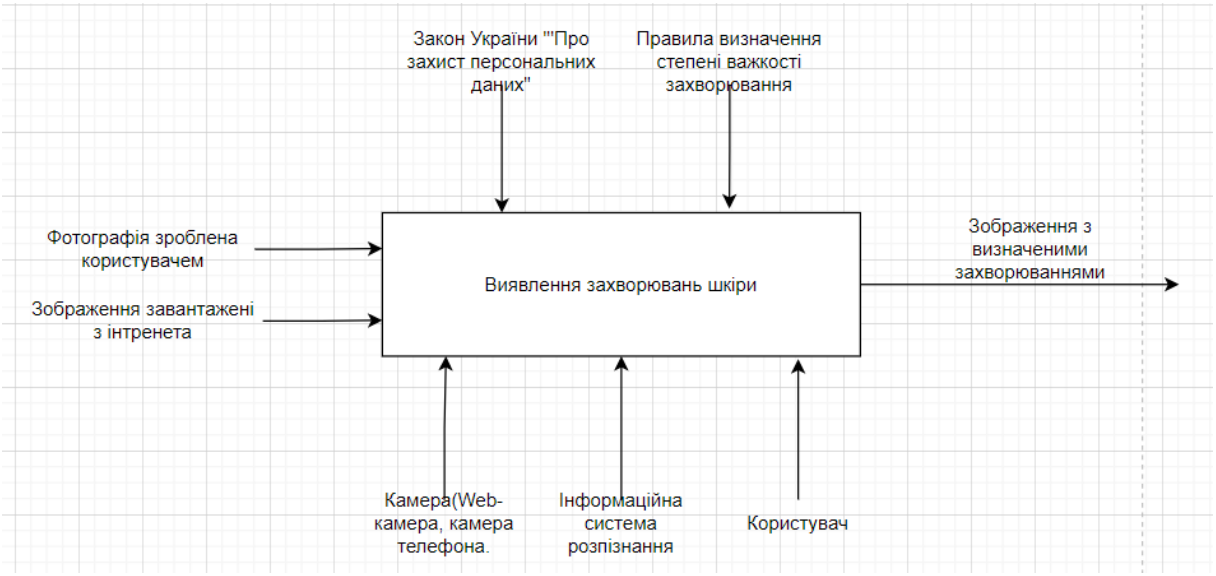


Рисунок 2.2 – IDEF0 ВЗШ (виявлення захворювань шкіри)

Опис стрілок (ICOM) системи:

1. Input – це об’єкти або дані, необхідні для виявлення захворювань шкіри: завантаження зображення з Інтернету, комп’ютер або телефон користувача, камери телефона або ПК, тощо.

2. Control – це правила, документи, які керують виконанням функції або визначають результати: закон України «Про захист персональних даних», правило виявлення важкості захворювання щкіри (фіксує ступінь ураження шкіри).

3. Mechanism – це особа або пристрій, які виконують функцію: користувач (дерматолог, косметолог, звичайний користувач тощо), який дає на вхід

зображення, та інформаційна система виявлення захворювань шкіри, яка визначить ступінь ураження шкіри.

4. Output – це дані, отримані як результат функції: ступінь ураження шкіри та зображення з місцями ураження.

2.3. Архітектура інформаційної системи

StarUML (Star Unified Modeling Language) є програмою для моделювання, яка дозволяє розробникам створювати діаграми, що представляють об’єкти, класи, компоненти, відносини та інші елементи програмного забезпечення за допомогою UML (Unified Modeling Language).

UML - це мова моделювання, яка дозволяє описувати структуру та поведінку програмного забезпечення. StarUML дозволяє розробникам створювати різноманітні діаграми UML, такі як діаграми класів, діаграми сеансів, діаграми послідовності, діаграми діяльності, діаграми станів тощо.

Для побудови архітектури додатку ВЗШ нам спочатку треба визначити компоненти системи в додатку StartUML. Додаток буде включати в себе клієнт-серверну архітектуру.

Клієнтський вузол («Client») буде складатися з таких компонентів:

− отримання введених користувачем даних («dataPackage»);

− взаємодія з сервером через сервер («clientNetwork»);

− опис користувацького інтерфейсу («clientGUI»).

Вузол серверної частини («Server») матиме три компоненти:

− підготовка вихідних даних («dataPreprocessing»);

− передобробка зображення («imagepreprocessing»);

− навчання нейронної мережі («trainingNN»);

− розпізнання («discernment»);

− бізнес-логіка сервера («businessLogic»);

− взаємодія з клієнтом через мережу («serverNetwork»).

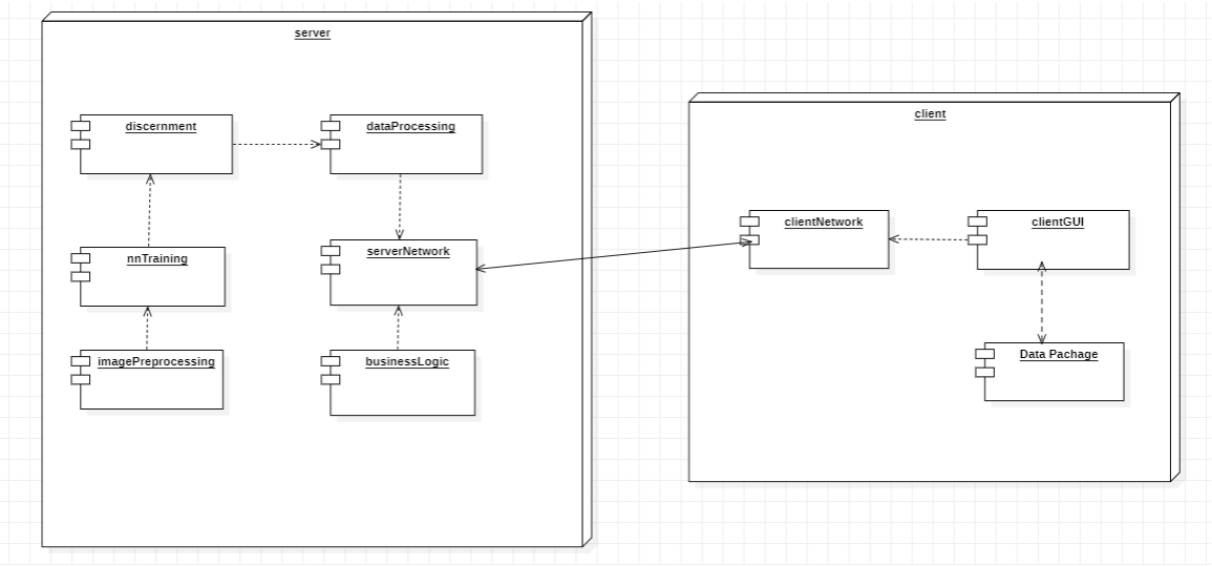


Рисунок 2.3 – Компонентна архітектура застосунку

Додаток складається з 2-х частин: «Client» і «Server».

«Client» містить два компоненти - clientGUI, який відповідає за користувацький інтерфейс, і «clientNetwork», що відповідає за взаємодію з сервером. Причому перший компонент залежить від другого. Client мав за мету зібрати вхідні дані від користувача («dataPackage») та вивести результат роботи.

«Server» містить в собі всю реалізацію серверної частини, «businessLogic» містить весь код, що реалізує бізнес-логіку сервера, «serverNetwork» реалізує взаємодію з клієнтом. «imagePreprocessing» відповідає за попередню обробку зображення, результат роботи, якої подається на вхід до «NN Training», де відбувається навчання нейронної мережі. Вхідні дані класифікуються «discernment» та передаються в «data Preprocessing», де вихідні дані передаються назад користувачеві.

2.4. Діаграма діяльності роботи застосунку

Перейдемо до описання основних життєвих циклів та їх переходів.

Побудуємо життєвий цикл розпізнаня захворювань шкіри (рис. 2.4):

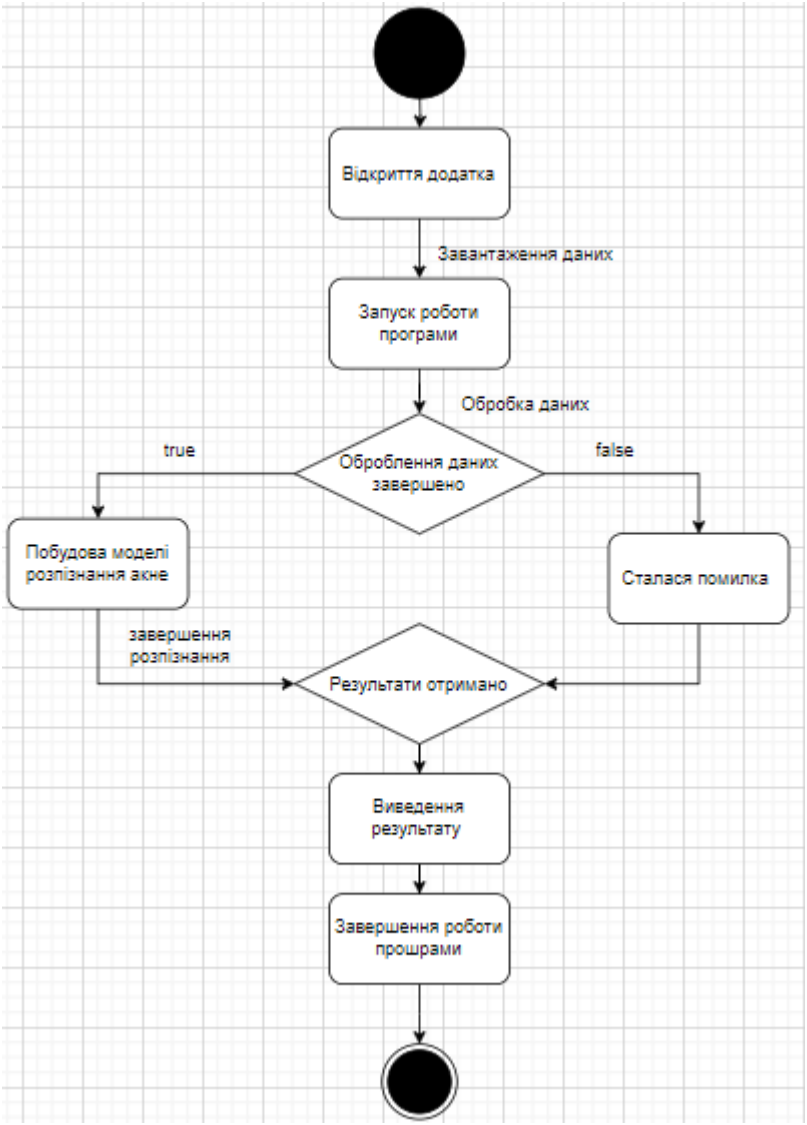


Рисунок 2.4 – Діаграма діяльності роботи застосунку

Побудуємо життєвий цикл моделі розпізнання (рис. 2.5).

Після завантаження програми, вводять вхідні дані. Вхідні дані користувача оброблюються і класифікуються системою, після чого надсилаються назад. Процес тренування майбутньої моделі для розпізнання захворювань шкіри матиме такі стани життєвого циклу:

− відкриття програми;

− завантаження даних;

− попередня обробка даних;

− налаштування параметрів моделі;

– навчання;

– зберігання моделі.

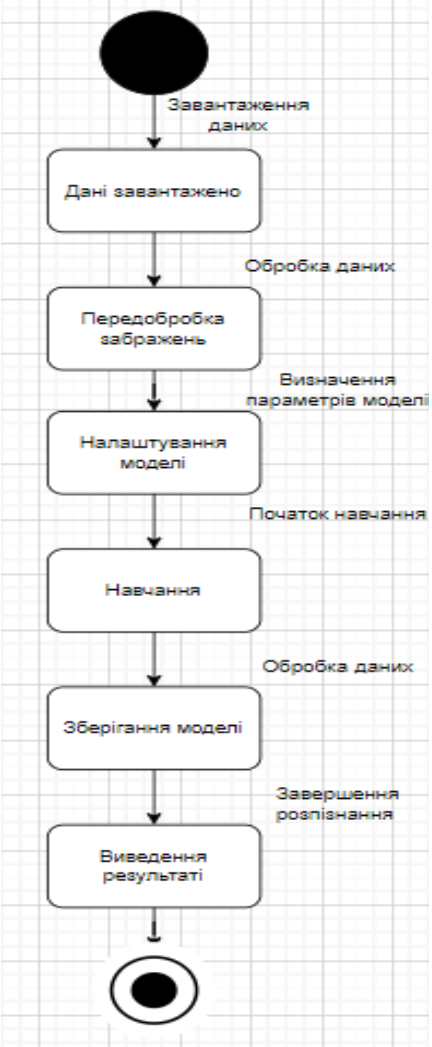


Рисунок 2.5 – Діаграма діяльності навчання моделі

Визначимо акторів, прецедентів та їх відношення системи. Як зазначалося раніше, в роботі беруть участь безпосередньо користувач та система.

Для користувача будуть доступні наступні дії:

– завантаження даних;

– отримання класифікаційної моделі;

– отримати результатів ВЗШ.

Для системи будуть характерними наступні дії:

– попередня обробка зображення;

– налаштування параметрів моделі;

− навчання моделі;

− виявлення захворювань шкіри.

З наведеного вище опису побудуємо UML діаграму (рис. 2.6).

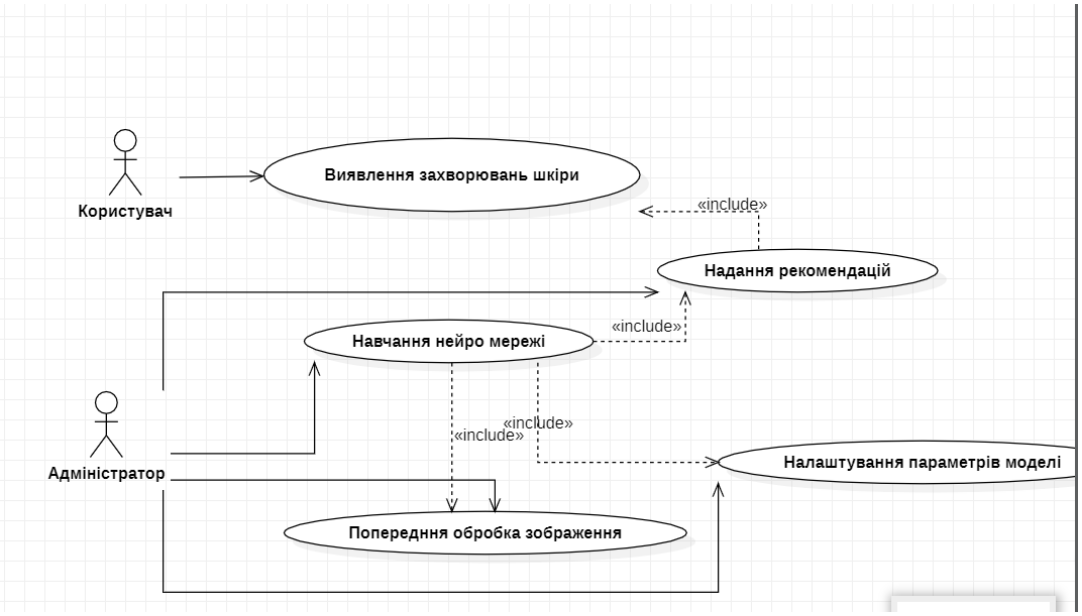


Рисунок 2.6 –Діаграма прецедентів розпізнавання акне

2.5. Побудова бізнес-моделі програмного модуля за допомогою EPC

Методологія ARIS базується на ієрархічній моделі бізнес-процесів, яка складається з різних рівнів деталізації. На найвищому рівні розташована стратегічна мета організації, наступні рівні включають такі елементи, як бізнес-процеси, функції, дії та ресурси.

EPC - Event-driven Process Chain, що означає ланцюжок бізнес-процесів, орієнтований на події. EPC, є графічною моделлю, яка використовується для моделювання бізнес-процесів в рамках методології ARIS (Architecture of Integrated Information Systems).

Головними елементами EPC є функціональні блоки, які представляють дії, що виконуються у рамках бізнес-процесу, та стрілки, які показують порядок виконання цих дій. Більш складні функціональні блоки можуть мати декілька

входів та виходів, що дозволяє відобразити різні умови виконання бізнес-процесу.

Діаграма бізнес-процесу в EPC повинна починатися і закінчуватися подією. Після функції завжди має відбуватись подія, тобто виконання функції створює деяка подія або стан. Бази даних, програмне забезпечення, документи тощо мають графічне позначення.

Діаграма бізнес-процесу описує:

– які дії виконуються в ході процесу;

– які організаційні підрозділи беруть участь у виконанні процессу;

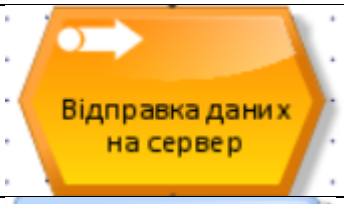
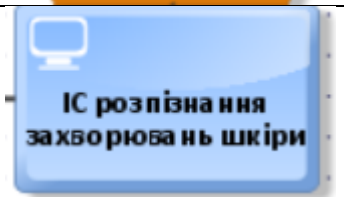
– які вхідні та вихідні дані використовуються;

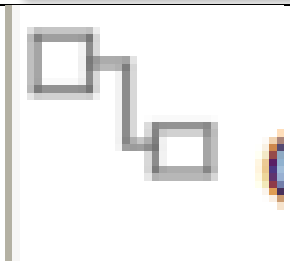
– які IT-системи задіяні;

– які події відбуваються в процесі.

Усі можливі варіанти робочого процесу представлені альтернативно за допомогою правил. Діаграма наведена у роботі побудована у нотації Event-Driven Process Chain у програмі ARIS Express. Далі представлена таблиця із умовними позначеннями, що містяться на діаграмі (табл. 2.1).

Таблиця 2.1 – Event-Driven Process Chain

	Activity − кроки, які необхідно виконати − описує, що відбувається під час процесу, тобто що саме робиться. Вони є основними елементами процесу. Приклад: завантажити зображення.
	Event − стан системи, процесу, що впливає на виконання функцій або керує ними. Наприклад: відправка даних на сервер.
	IT-system (IT-система) − посилання на EDP (electronic data processing): дії можуть виконуватися вручну або автоматично. Автоматизовану діяльність здійснюють IT-системи. Приклад: AWS/EC2.

	Role – розподіл ролей для діяльності – ілюструє, хто виконує діяльність. Приклад: адміністратор.
	Потік даних – потік даних у процесі: вхідні та вихідні дані. Процес генерує дані або вимагає даних для продовження. Ці дані моделюються як вхідні або вихідні дані діяльності.
	Правила – описують альтернативи робочого процесу і таким чином ілюструють можливі варіанти виконання. Доступні такі символи правил: АБО: можна використовувати будь-яку кількість шляхів. І: усі наступні/попередні шляхи застосовуються. Виключаюче АБО: може мати місце лише один з наступних/попередніх варіантів (як у цьому прикладі: або дані обробилися коректно, або виникла помилка).

1. Ролі:

1.1. Користувач (відвідувач веб-сайту: косметолог, дерматолог, звичайний користувач, зацікавлена особа).

1.2. Адміністратор (налаштовує параметрі моделі і відповідає за функціональність серверу).

2. Діяльність:

2.1. Відкрити веб-сайт (користувач вводить в пошуковому рядку назву сайту, атрибут ВЗШ).

2.2. Завантажити зображення (користувач завантажує одне зображення з комп’ютера, мобільного телефону чи стаціонарної камери).

2.3. Обробка даних (розпізнавання та підрахунок акне на сервері).

2.4. Відправити дані користувачу (вивести на екран користувача оброблені зображення, яке містить захворювання шкіри).

2.5. Відправити повідомлення про помилку (виведення в консолі адміністратора повідомлення про помилку в момент, коли програма дала збій).

2.6. Повторне введення даних (користувач може вести нові дані, або коригувати попередні).

3. Події:

3.1. Сайт вдало завантажився (веб-сайт готовий для використання).

3.2. Сталася помилка (користувач має проблеми зі своїм пристроєм, маршрутизатором чи провайдером, є загроза наявності фішингової сторінки або вірусу тощо).

3.3. Зображення завантажено (завантажено вибрані користувачем зображення, які вдало додалися до системи).

3.4. Відправка даних на сервер (зображення відправляються на сервер для обробки).

3.5. Дані оброблено коректно (введені користувачем зображення задовільні).

3.6. Сталася помилка при обробці (технічні причини, наприклад, проблеми з провайдером, або з DNS).

3.7. Вивід результатів на екран (вивести на екран користувача оброблені зображення, які містять виявлені захворювання шкіри)

3.8. Користувач бажає ввести нові дані (користувач може ввести нові дані, або коригувати попередні).

3.9. Виявлення захворювань шкіри завершено (користувач закінчив роботу).

4. ІТ-система:

4.1. ІС виявлення захворювань шкіри (знаходиться на Google colab, зберігає у собі інформацію навченою нейронною моделлю).

5. Зовнішні об'єкти:

5.1. Стаціонарна камера, камера мобільного телефону (користувач може використовувати утиліти для завантаження зображення)

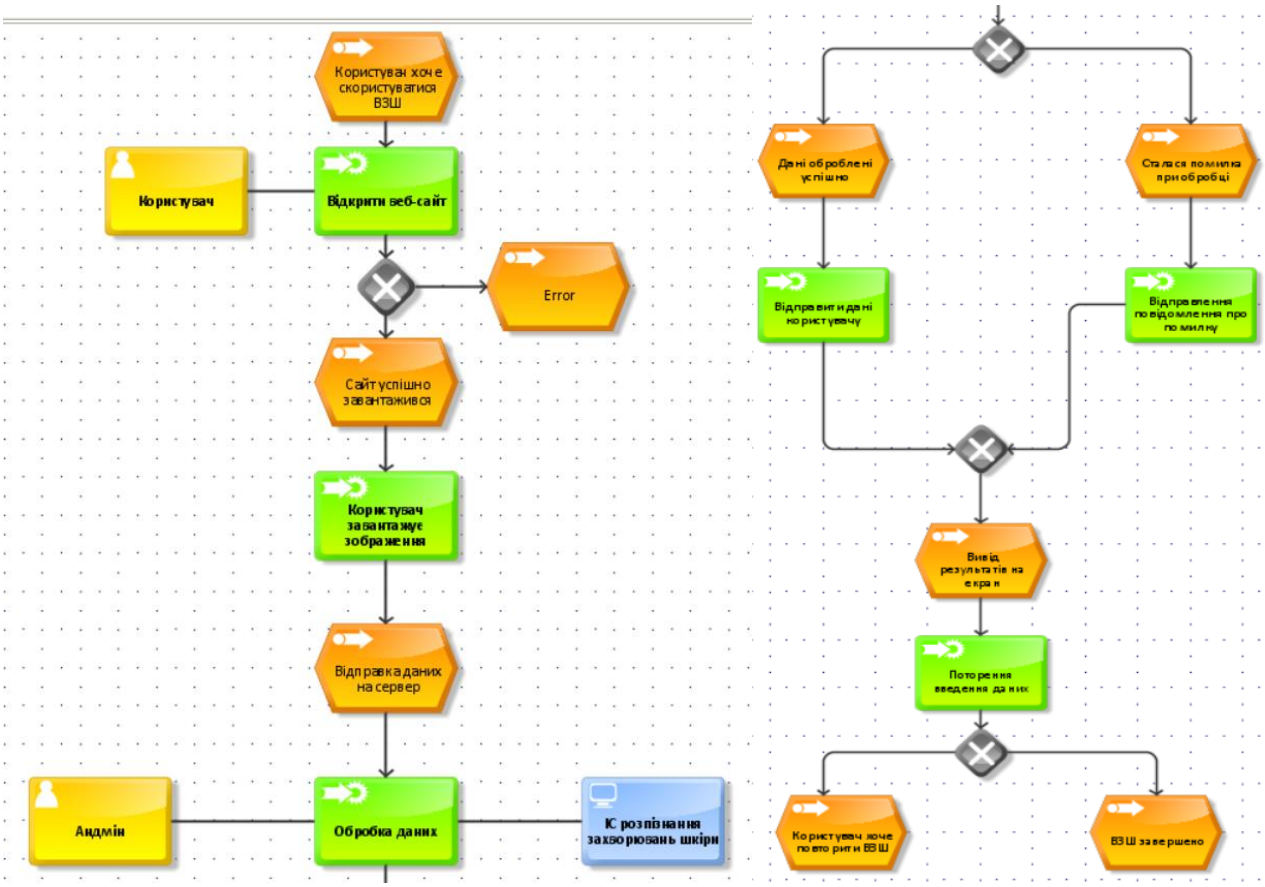


Рисунок 2.8 - Діаграма бізнес-процесу в EPC

2.6. Фізична архітектура системи

Папки:

– hooks – папка з кастомними хуками

– pages – папка з сторінками сайта

– public – коренева папка проекту

– state – папка зі станами компонентів додатка

– style – папка з стилями сайта

– utils – містить в собі імпорт клінта supaBase

– api – містить в собі підключиння до back-end та базу даних.

JS скрипти:

acne.js – формування POST на модель по виявленню захворювань шкіри

app.js – корнева папка пректу

ImageWithLabels.js – компонент для відображення міток

Predictionmap.jsx – компонент для відображення вітсотка впевненості в захворюванні шкіри

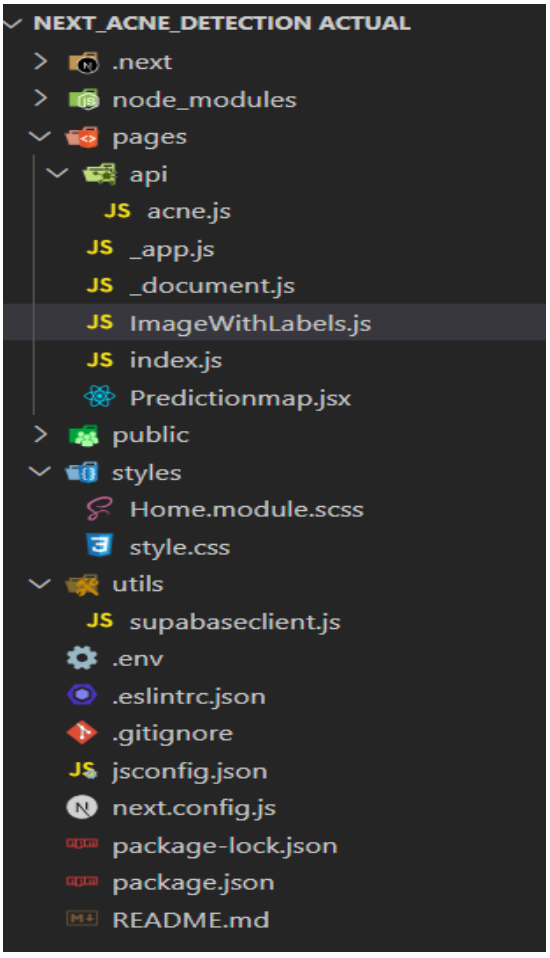


Рисунок 2.9 - Фізична архітектура системи

2.7. Вибір середовища та інструментів реалізації для розпізнання захворювань шкіри

1. Roboflow

Roboflow [21] - це платформа комп’ютерного зору, яка надає інструменти для розробників і бізнесів для створення, навчання та розгортання моделей комп’ютерного зору. Платформа пропонує різноманітні функції, такі як анотування даних, попередня обробка зображень, тренування моделей та розгортання, а також інтеграцію з популярними фреймворками глибокого навчання, такими як TensorFlow, PyTorch та Keras. Roboflow має на меті спростити процес створення моделей комп’ютерного зору, надаючи зручну та доступну платформу.

Roboflow має кілька переваг порівняно зі своїми конкурентами:

1.Велика кількість передньо навчених моделей: Roboflow пропонує більше 50 моделей, що дає можливість вибрати найкращу для певної задачі, що робить платформу більш гнучкою та зручною для використання.

2.Підтримка різних форматів даних: Roboflow підтримує більше 30 різних форматів даних, що дає можливість використовувати дані з різних джерел.

3.Зручний інтерфейс та документація: Roboflow пропонує детальну документацію та зручний інтерфейс, який робить процес розробки моделей комп’ютерного зору менш складним та більш доступним для початківців.

4.Підтримка широкого спектру задач: Roboflow підтримує різні типи задач комп’ютерного зору, такі як виявлення об’єктів, класифікація зображень, сегментація тощо.

5.Гнучкість та масштабованість: Roboflow дає можливість користувачам використовувати власні дані та налаштовувати параметри моделей, що робить платформу більш гнучкою та масштабованою.

2. YOLO

YOLO (You Only Look Once) - це алгоритм для виявлення об’єктів на зображеннях або відео. Відрізняється від більш традиційних алгоритмів, які розділяють обробку зображення на кілька етапів, таких як виявлення областей інтересу, виділення ознак та класифікація. У YOLO процес виявлення об’єктів

відбувається в один прохід, що робить його значно швидшим та менш вимогливим до ресурсів.

Алгоритм YOLO використовує нейронну мережу, яка поділяє вхідне зображення на сітку та для кожної клітинки сітки передбачає, які об’єкти знаходяться в ній, і який їх клас, а також їхні координати. Таким чином, YOLO здатний виявляти більше одного об’єкту на зображенні та відрізняти їх від фону.

3. Next js

Next.js - це фреймворк для розробки веб-додатків на основі React. Він надає розробникам зручний і простий спосіб створення універсальних та серверних додатків, що можуть бути виконані як на клієнті, так і на сервері.

Основна перевага Next.js полягає в тому, що він автоматично оптимізує швидкість завантаження веб-сторінок, включаючи попередній рендеринг (pre-rendering), що дозволяє відображати веб-сторінки більш швидко та зменшує навантаження на сервер.

Крім того, Next.js дає можливість використовувати серверні файли з базами даних та API-інтерфейсами, що дозволяє розробникам створювати повноцінні веб-додатки зі складним функціоналом.

4. SupaBase

Supabase - це відкрите хмарне рішення для розробки баз даних та серверних додатків. Вона надає розробникам можливість швидко створювати та масштабувати повноцінні серверні додатки з використанням баз даних PostgreSQL. Основною метою Supabase є забезпечення зручного та ефективного способу створення додатків з повноцінною серверною логікою та базою даних без необхідності власноруч встановлювати та керувати інфраструктурою.

Ось кілька переваг Supabase:

1.Відкрите програмне забезпечення: Supabase є відкритим проектом, що означає, що ви можете переглядати та модифікувати його вихідний код. Ви

можете розгортати Supabase на своїх серверах або використовувати готовий хмарний сервіс.

2.PostgreSQL-сумісність: Supabase базується на базі даних PostgreSQL, яка є однією з найпотужніших та надійних систем керування базами даних. Ви можете використовувати всі можливості PostgreSQL, такі як реляційні операції, індекси, транзакції та багато іншого.

3.Real-time оновлення: Supabase надає вбудовану підтримку реального часу, що дозволяє синхронізувати дані між клієнтами та сервером у реальному часі. Це особливо корисно для розробки колаборативних додатків, чатів, стрімінгових сервісів тощо.

4.Автентифікація та авторизація: Supabase надає механізми автентифікації та авторизації, які дозволяють легко управляти доступом до вашого додатку. Ви можете налаштувати різні методи аутентифікації, такі як електронна пошта та пароль, соціальн

Висновок другого розділу

За результатами виконання другого розділу кваліфікаційної роботи було проведено: функціональний аналіз, побудову дерева функцій. Було показано узагальнену технічну архітектуру системи, визначено підсистеми, модулі та зв’язки між компонентами системи. Сформовано архітектуру системи у нотації IDEF0 та побудовано бізнес-моделі програмного модуля за допомогою EPC.

Розроблено структуру та головну функціональність клієнт-серверного застосунку із використання мови програмування Python, бібліотек обробки зображень OpenCV та Pillow, бібліотеки Streamlit для візуалізації інтерфейсу користувача. Розгорнуто серверну частину додатку на хмарній платформі Amazon Web Services.

РОЗДІЛ 3. ТЕХНОЛОГІЧНІ ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ ПРОГРАМНОГО МОДУЛЯ

3.1. Реалізація програмного модулю розпізнання захворювань шкіри

3.1.1. Створення та експериментальні верифікації нейронної мережі для розпізнавання акне

Оскільки roboflow приймає набор даних у форматі img-labels, то для тренування нейромережі використовуємо зображення з відредагованого набору даних та файли JSON, які містить деталі зображення: назва, координати обмежувальних боксів, складність виявлення (рис. 3.1).


```
{
  "predictions": [
    {
      "x": 301.5,
      "y": 68,
      "width": 19,
      "height": 18,
      "confidence": 0.675,
      "class": "acne"
    },
    {
      "x": 168.5,
      "y": 364.5,
      "width": 15,
      "height": 19,
      "confidence": 0.653,
      "class": "acne"
    },
```


Рисунок 3.1 – Формат набору даних для roboflow

Мережі та набори даних у roboflow можна перемикати та навчати уніфікованим способом, щоб точно налаштувати та порівняти різні мережі. У

ході експериметальних верифікацій найкращу ефективність показав метод Faster R-CNN Inception-ResNet-v2 (табл. 3.1):

Таблиця 3.1. Порівняння різних типів нейроних мереж у roboflow

Тип нейромережі	Витрати пам'яті (GB)	Загальний час тренування (год)	Точність
YOLOv5	0,25	2	0,75
EfficientDet	1	12	0,86
EfficientDet	0,10	1	0,72
Mask R-CNN	2	15	0.82
Roboflow 2.0 object detection	1	5	0,88

Roboflow 2.0 Object Detection - це оновлена версія платформи для розпізнавання об’єктів на зображеннях. Вона дозволяє швидко і ефективно створювати, тренувати та розгортати моделі для розпізнавання об'єктів на основі зображень.

Процес роботи з обробки зображень в Roboflow 2.0 Object Detection може бути поділений на кілька етапів:

1. Завантаження даних: спочатку необхідно завантажити дані в платформу. Для цього можна використовувати різні формати зображень, такі як JPEG, PNG або BMP.

2. Перетворення даних: після завантаження даних необхідно виконати їх попередню обробку та підготовку для навчання моделі. Roboflow 2.0 Object Detection автоматично виконує ці операції, включаючи зменшення розміру зображень, їх розміщення в форматі, зрозумілому для моделі, а також генерування додаткових зображень за допомогою аугментації.

3. Вибір архітектури мережі: після підготовки даних необхідно вибрати архітектуру мережі, яку будемо використовувати для навчання моделі. Roboflow 2.0 Object Detection містить кілька зареєстрованих архітектур, які можна використовувати.

4. Тренування моделі: після вибору архітектури мережі необхідно розпочати навчання моделі. Для цього використовуються стандартні методи навчання, такі як зворотне поширення помилки та оптимізатори градієнта.

5. Оцінка моделі: після завершення тренування моделі необхідно оцінити її ефективність на тестових даних. Для цього використовуються метрики, такі як точність, чутливість та специфічність.

6. Підбір гіперпараметрів: для покращення ефективності моделі може бути необхідно підібрати оптимальні значення гіперпараметрів, такі як швидкість навчання та кількість епох.

Параметри навчання підбиралися експериментальним шляхом:

Кількість епох навчання – 384. Значення даного коефіцієнта визначається кількістю повних проходжень тренувальних даних через нейромережу. За одну епоху відбувається кілька ітерацій. Їх кількість визначається числом вхідних знімків (284 зображення жовтої стрічки) та розміром батча.

Дата сет був сформований власноруч .Зображення взяті з kaggle , мітки були розставлені. Та сформований дате сет з 975 зображень

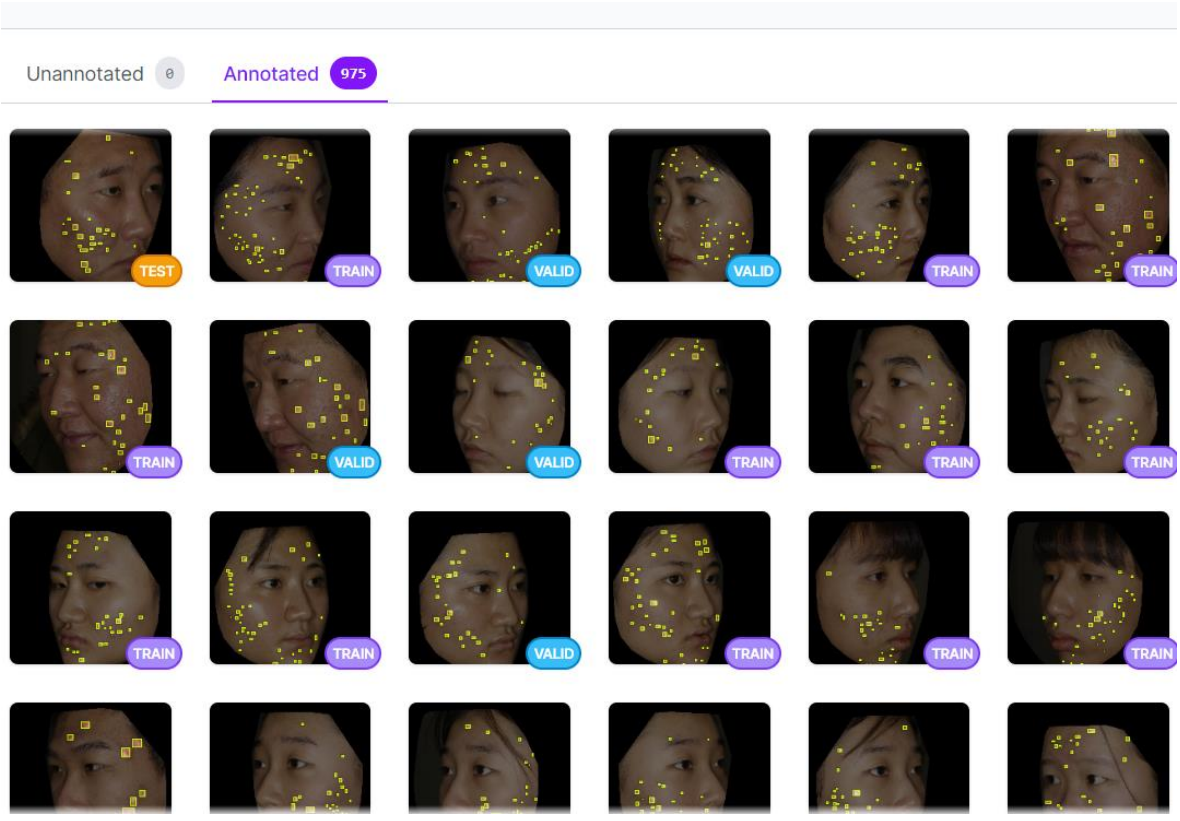


Рисунок 3.2 – Дата сет

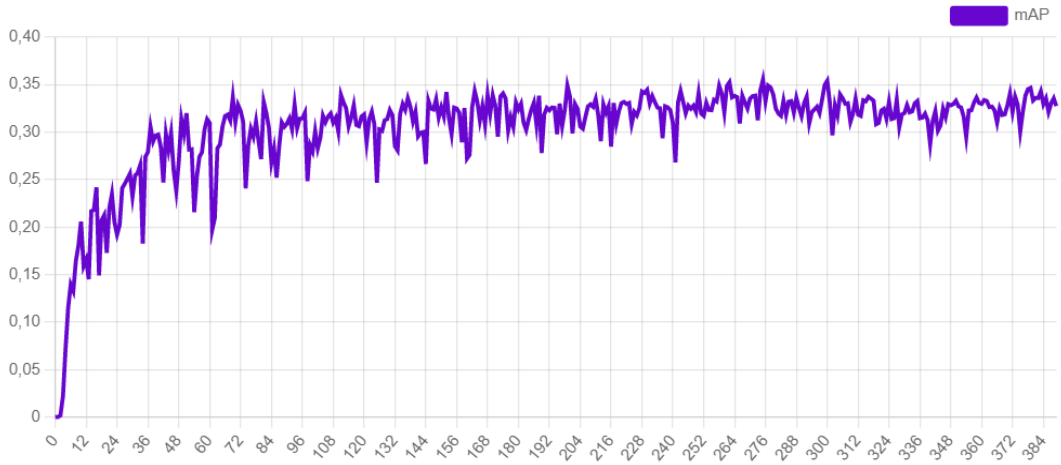


Рисунок 3.3 – Графіки зміни точності створеної моделі

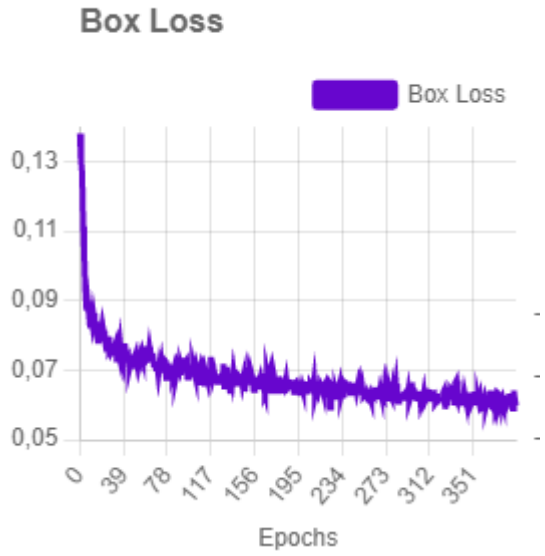


Рисунок 3.4 – Графіки втрата рамки

1. Box loss (втрата рамки) - це функція втрат, яка використовується в моделі об'єктного виявлення для підсумовування помилок в прогнозах рамок, які були зроблені моделлю під час навчання.

Рисунок 3.5 – Графіки втрата об`єкта

2. Object loss (втрата об`єкта) - це функція втрат, яка використовується в моделі об'єктного виявлення для підсумовування помилок в прогнозах наявності об'єктів на зображенні, які були зроблені моделлю під час навчання.

Рисунок 3.6 – Графіки втрата класу

З графіків можна зрозуміти, що параметри нейронної мережі були належним чином налаштовані. Новостворена модель буде протестована на перевірочному наборі даних (рис. 3.4).

Рисунок 3.7 – Тестування створеної нейромережі (до обробки)

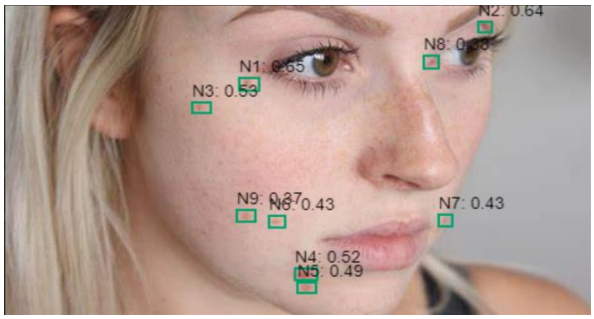


Рисунок 3.8 – Тестування створеної нейромережі (після обробки)

3.1.2. Реалізація основного модулю у Acne_detection

При натискані на кнопку submit та виборі користувачем зображення, яке він хоче обробити. Це зображення записується в БД supabase [18], для того, щоб далі передати його у вигляді url в нашу навчану модель нейромережі.

Після чого ми повинні отримати наше зображення у вигляді url в клієнтській частині.

Лістинг 1 – отримання зображення у вигляді url

```
const { data: list, error: err } = await supabase.storage
      .from("acne")
      .list("imgs");
    const filesToRemove = list.map((x) => `imgs/${x.name}`);
    const { data: data_acne, error: erroe_acne } = await supabase.storage
      .from("acne")
      .remove(filesToRemove);
    const { error } = await supabase.storage
      .from("acne")
      .upload(`imgs/${imageName}`, imageFile, {
        cacheControl: "3600",
        upsert: false,


      });
```

Після того як ми отримали наше зображення у вигляді url нам треба з клієнської частини зробити запит на сервер, де в нас знаходиться використована нейромережа.

Лістинг 2 – відправка запиту на сервер

```
Axios({
  method: "POST",
  url: "https://detect.roboflow.com/acne_detection/1",


  params: {
    api_key: "",
    image: publicUrl,
    confidence: 30,
    format: "JPEG",
  },
})
```

Ми робимо POST запит в якому вказуємо в url- шлях до нашої нейромережі. Потім ми вказуємо параметри:

1. Api_key – ключ нашої моделі
2. image – наше зображення в url вигляді

3. confidence – мінімальне значення впевненості для моделі

4. Формат зображення , яке ми отримаємо назад

Після чого ми отримуємо відповідь серевера у вигляді JSON в якому знаходиться масив об’єків

У такому вигляді ми отримуємо відповідь від сервера – це масив об’єктів кожен з цих об’єктів складається з наступних полів :

1. x – координати мітки по x

2. y – координати мітки по y

3. width – ширина мітки

4. heigth – висота мітки

Після чого нам потрібно вказати по цьому масиву об’єктів наші мітки.

Я створив окремий компонент ImageWithLabels, в який передав наступні параметри :

1. imageWidth – ширина картинки

2. imageHeight – висота картинки

3. imageSize – загальний розмір картинки

4. labelX – розташування мітки по x

5. labelY – розташування мітки по y

6. labelWidth – ширина мітки

7. labelheight – вистота мітки

8. labelConfidence – впевненість розпізнання.

За цими параметрами вказую labels.

Лістинг 3– створення міток на фото

```
function drawRect(ctx, x, y, w, h) {
  ctx.beginPath();
  ctx.rect(x, y, w, h);
  ctx.strokeStyle = "#02a16c";
  ctx.lineWidth = 2;
  ctx.stroke();
}
```

3.2 Інструктивний матеріал з експлуатації виявлення захворювань шкіри

3.2.1 Інструктивний матеріал з експлуатації виявлення захворювань шкіри для користувача

Завдяки створенню web-додатка було максимально спрощено взаємодію додатка та користувачів (косметолог, дерматолог), які повинні просто зайти на веб-сайт.

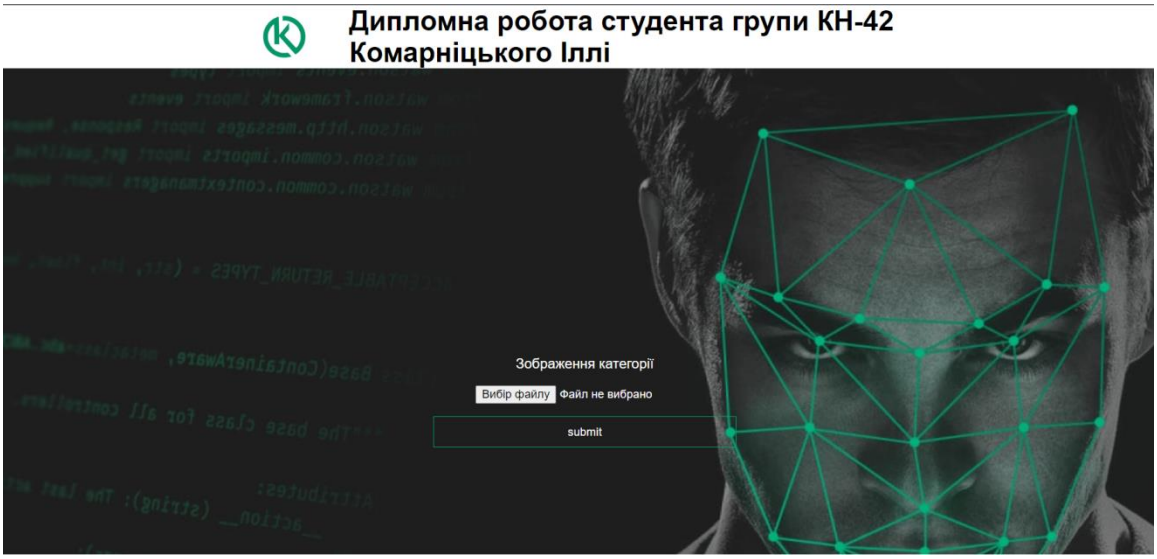


Рисунок 3.9– Вигляд web-додатка

Для того щоб здійснити виявлення акне, необхідно натиснути на кнопку «Вибір файлу». Після чого вибрати зображення та натиснути на кнопку «submit».

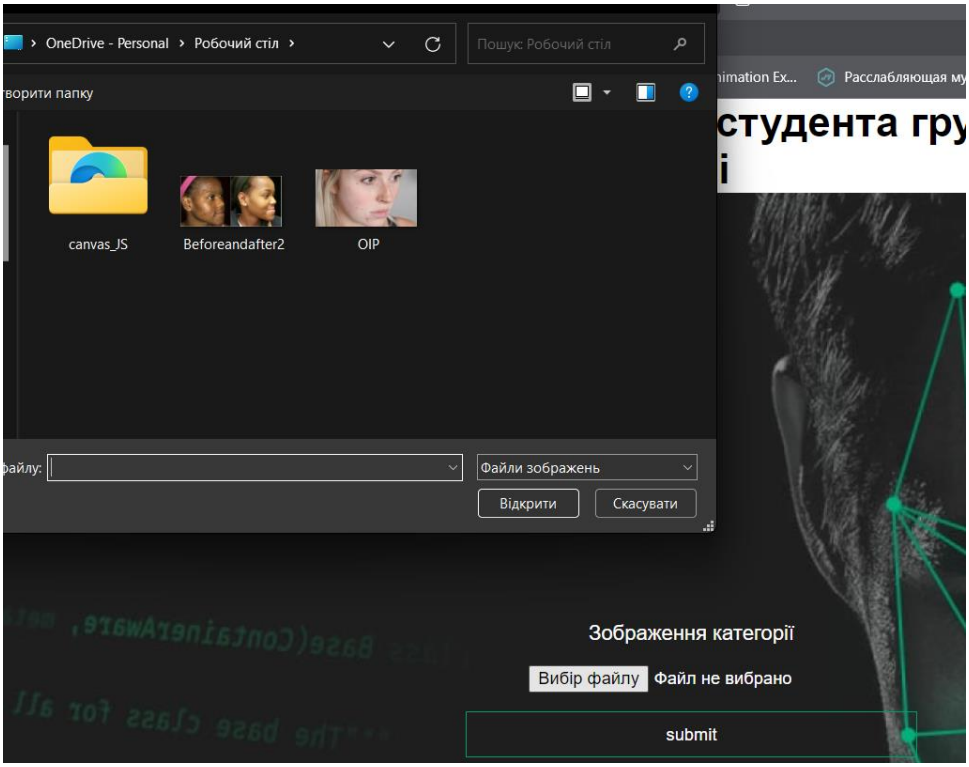


Рисунок 3.10– Вибір зображення

Після того як обрано зображення можна побачити результат роботи програмного модулю.

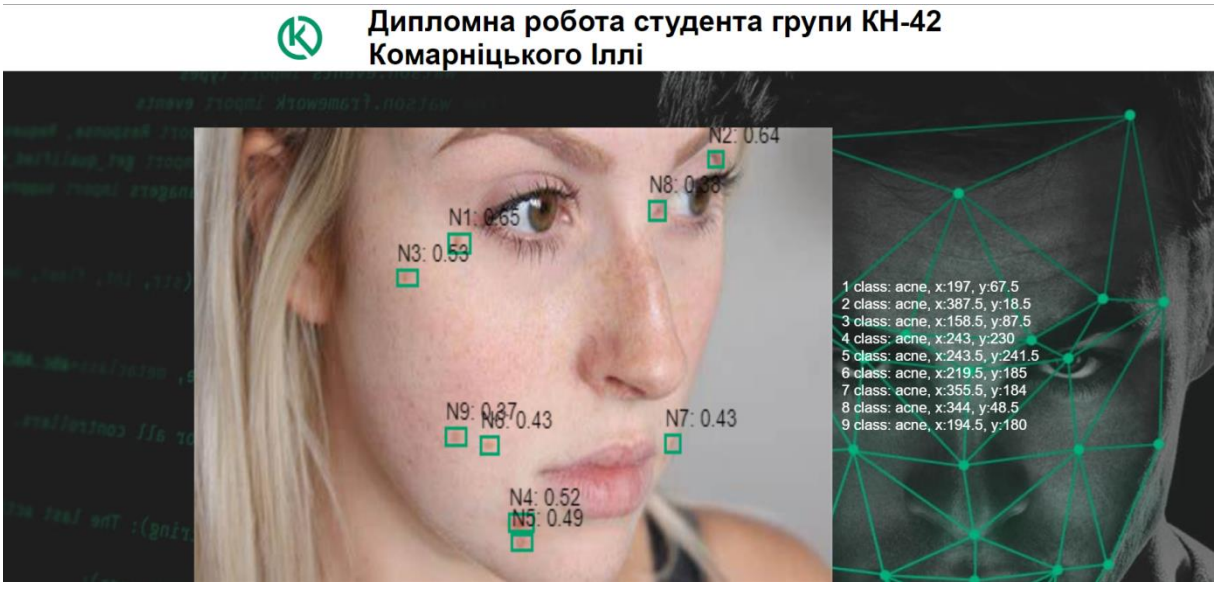


Рисунок 3.11 – Виконання прогнозу

3.3. Тестування та аналіз модулю виявлення захворювань шкіри

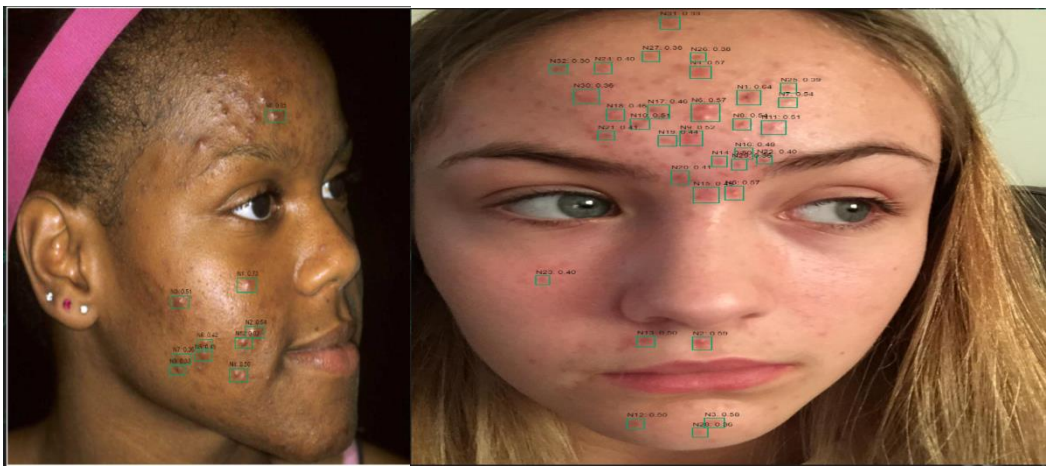
Розглянемо можливі випадки, які можуть трапитися під час виявлення акне. Виділимо наступні випадки:

1. Якщо при виявленні захворювань шкіри на зобрженні буде здорова людина та хвора.

Рисунок 3.12 – Виконання прогнозу на декількох людях

Результат: ці фото демонструють, що програма виконує свою роботу належним чином, виставляючи мітки тільки на фото хворих людей. Це підтверджує правильність навчання моделі.

2. Перевірка роботи програми на людях різної раси



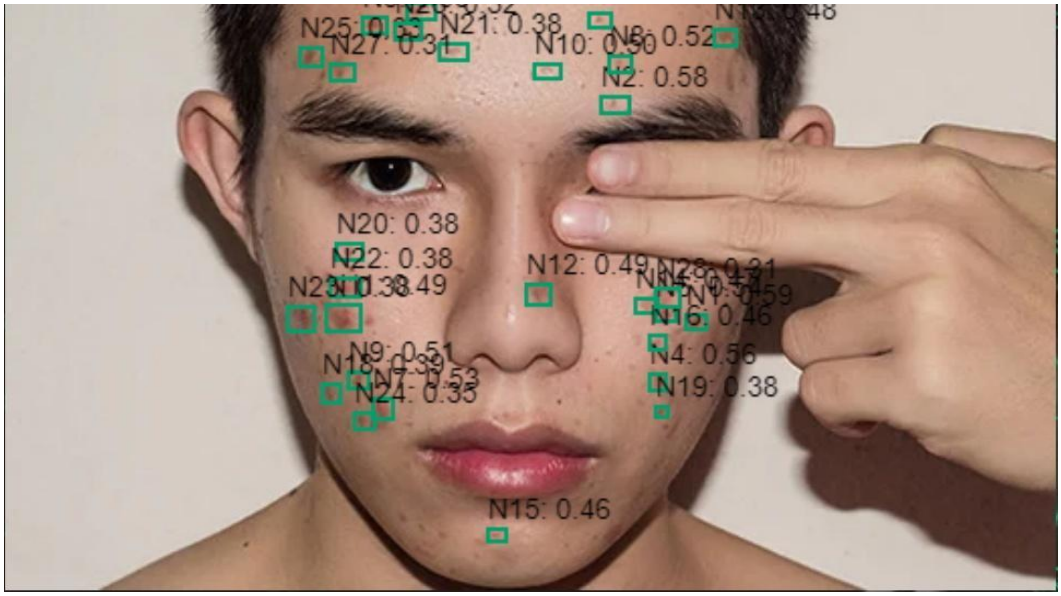


Рисунок 3.13 – Виконання прогнозу на людях різної раси

Результат: отримане зображення свідчить про правильну роботу програмного коду на людях різних рас, що демонструє використання датасету, що охоплює представників різних рас.

3. Перевірка роботи на фотографіях людей, зроблених за допомогою телефону.

Рисунок 3.14 – Виконання прогнозу на фотографіях людей, зроблених за допомогою телефону

Рисунок 3.15 – Виконання прогнозу на фотографіях людей, зроблених за допомогою телефону

Результат: видно, що модель ефективно функціонує, якщо застосовані фотографії без фільтрів і не мають наслідків від пост-акне. Це свідчить про те, що програма коректно працює навіть з фотографіями, зробленими на мобільному телефоні.

4. Перевірка роботи програми на зображеннях в темноті.

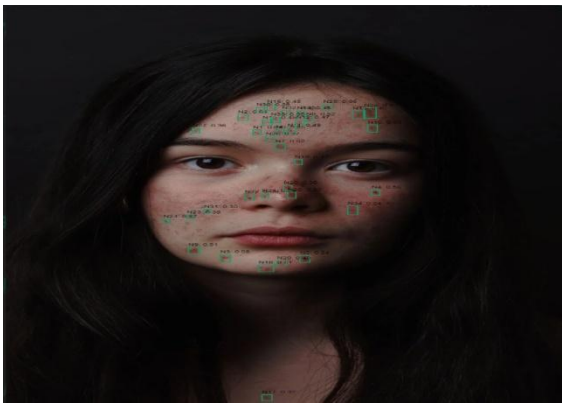


Рис 3.16 – Виконання прогнозу вночі

Рис 3.17 – Виконання прогнозу вночі

Результат: модель вірно розпізнає зображення навіть в умовах недостатнього освітлення, що свідчить про її здатність ефективно працювати як з яскравими, так і з затемненими зображеннями.

Висновок третього розділу

У третьому розділі кваліфікаційної роботи наведено струтуру розробленого web-додатка та моделі для розпізнання захворювань шкіри. Описано кожну частину проекту та детально розібрано програмний модуль (клієнська частина та серверна). Приведені всі бібліотеки та компоненти, які використовувались в проекті.

Наведено детальну інструкцію для користувача щодо використання web-додатку: користувацький інтерфейс та робота із застосунком. Проведено тестування програмного модулю для різних випадків. Виявлено як слабкі, так і сильні сторони моделі. Тестування проводилося в різноманітних ситуаціях та за різних умов.

Результати вказують на те, що програма виконує свою роботу належним чином, виставляючи мітки тільки на фото хворих людей, що підтверджує правильність навчання моделі. Отримані зображення свідчать про правильну роботу програмного коду на людях різних рас, що демонструє використання датасету, охоплюючого представників різних рас. Модель також ефективно функціонує з фотографіями без фільтрів та наслідків пост-акне, включаючи

зображення, зроблені на мобільному телефоні. Крім того, модель правильно розпізнає зображення навіть в умовах недостатнього освітлення, демонструючи її здатність ефективно працювати як з яскравими, так і з затемненими зображеннями.

ВИСНОВКИ

У кваліфікаційній роботі наведено опис області застосування детектора акне, аналітичний огляд методів захворювань шкіри, виявлення зацікавлених сторін проекту.

Розглянуто наявні рішення (існуючі сервіси з аналізу обличчя на акне), представлено короткі теоретичні відомості про можливі важкості акне, що будуть використовуватись у програмі, а також детально описана постановка задачі.

Проведено функціональний аналіз та визначено вимоги до програмного модулю, розроблено архітектуру застосунку, бізнес-логіку, підсистеми, модулі та зв’язки між компонентами системи. Додаток було розроблено з використанням мови програмування Js та бібліотеки nextJS.

Наведено струтуру розробленого web-додатку та моделі для розпізнання захворювань шкіри. Описано кожну частину проекту та детально розібрано програмний модуль (клієнська частина та серверна). Приведені всі бібліотеки та компоненти, які використовувались в роботі.

Наведено детальну інструкцію для користувачів щодо використання web-додатку: користувацький інтерфейс та робота із застосунком. Вони мають можливість приєднатись до процесу навчання класифікаційної моделі з метою покращання її точності виявлення acne. Проведено тестування програмного модулю для різних випадків.

Розроблений програмний модуль дає можливість виявляти захворювання шкіри обличчя з метою його подальшого лікування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Barzegar, M., Safari, A., & Dehghan, M. (2018). Skin lesion segmentation using convolutional neural networks and thresholding techniques. Computer methods and programs in biomedicine, 156, 55-68.
2. Bengio, Y. (2009). Learning deep architectures for AI. Foundations and Trends® in Machine Learning, 2(1), 1-127.
3. Cruz-Ramírez, N., Carrasco-Ochoa, J. A., & Martínez-Trinidad, J. F. (2019). Ensemble of convolutional neural networks for skin lesion classification. Expert Systems with Applications, 131, 208-221.
4. Deng, L., & Yu, D. (2014). Deep learning: methods and applications. Foundations and Trends® in Signal Processing, 7(3-4), 197-387.
5. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
6. Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.
7. Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700-4708).
8. Jena, P. K., & Panda, R. (2020). An automated system for acne detection and classification using deep learning techniques. Neural Computing and Applications, 32(9), 3923-3937.
9. Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260.
10. Li, Y., Liu, L., & Tian, Q. (2020). Deep learning for skin lesion diagnosis: A survey. IEEE Transactions on Medical Imaging, 39(9), 2654-2664.

11. Ranjan, R., & Singh, V. (2019). Skin lesion classification using deep neural network: A review. Computer Methods and Programs in Biomedicine, 175, 227-237.
12. Sharma, P., & Thakur, A. (2020). Skin Lesion Segmentation using Fully Convolutional Neural Network. International Journal of Scientific Research in Computer Science and Engineering, 8(1), 9-13.
13. Song, E., Kim, W., Jeong, J., & Kim, H. G. (2019). A deep learning-based approach for melanoma detection using dermoscopy images. Computer Methods and Programs in Biomedicine, 179, 104998.
14. Yang, H., Liu, Z., & Zhang, H. (2021). Automated Skin Lesion Diagnosis Based on Deep Learning Techniques: A Survey. IEEE Access, 9, 128535-128551.
15. skinvisionary [Електронний ресурс] – Режим доступу до ресурсу: https://www.designerskin.com/catalog/intensifier/designer-skin-visionary (дата звернення: 19.04.2023).
16. SkinVision [Електронний ресурс] – Режим доступу до ресурсу: https://www.skinvision.com/gettingstarted (дата звернення: 19.04.2023).
17. MDacne [Електронний ресурс] – Режим доступу до ресурсу: https://www.mdacne.com/how_it_works (дата звернення: 19.04.2023).
18. Supabase [Електронний ресурс] – Режим доступу до ресурсу: https://supabase.com/ (дата звернення: 19.04.2023).
19. React [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/ (дата звернення: 19.04.2023).
20. NextJs [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/ (дата звернення: 19.04.2023).
21. Roboflow [Електронний ресурс] – Режим доступу до ресурсу: https://roboflow.com/ (дата звернення: 19.04.2023).
22. Yollo5 [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/ultralytics/yolov5 (дата звернення: 19.04.2023).
23. Kaggle [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/ (дата звернення: 19.04.2023).

24. Scss [Електронний ресурс] – Режим доступу до ресурсу: https://sass-lang.com/documentation/syntax (дата звернення: 19.04.2023).
25. Tensorflow [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/ (дата звернення: 19.04.2023).

ДОДАТКИ

Додаток А

Лістинг програмного коду

```
import Head from "next/head";
import Image from "next/image";
import { Inter } from "next/font/google";
import styles from "../styles/Home.module.scss";
import { useState, useRef, useEffect } from "react";
import { supabase } from "@/utils/supabaseclient";
import { useRouter } from "next/router";
import axios from "axios";
import ImageWithLabels from "./ImageWithLabels";
import Predictionmap from "./Predictionmap";
import img from "../public/logo.jpg";

const inter = Inter({ subsets: ["latin"] });

export default function Home({ acne }) {
  const [imageFile, setImageFile] = useState(null);
  const imageFileRef = useRef(null);
  const router = useRouter();
  const [data, setData] = useState("");
  const [imageUrl, setImageUrl] = useState("");
  // console.log(acne);
  const uploadImage = async () => {
    const imageName = `${Date.now()}-${imageFile.name}`;

    const { data: list, error: err } = await supabase.storage
      .from("acne")
      .list("imgs");
    const filesToRemove = list.map((x) => `imgs/${x.name}`);
    const { data: data_acne, error: erroe_acne } = await supabase.storage
      .from("acne")
      .remove(filesToRemove);
    const { error } = await supabase.storage
      .from("acne")
      .upload(`imgs/${imageName}`, imageFile, {
        cacheControl: "3600",
        upsert: false,
      });

    if (error) return alert(error.message);

    // const { data } = await supabase.storage
    //   .from("acne")
    //   .getPublicUrl(`imgs/${imageName}`);

    const { data } = await supabase.storage
      .from("acne")
      .getPublicUrl(`imgs/${imageName}`);

    // console.log(new URL(data.publicUrl).href);
    return new URL(data.publicUrl).href;
  };

  const handleImageChange = (e) => {
    const image = e.target.files[0];
```

```
  if (image.size > 2000000) {
    alert("Image size must be less than 2MB");
    return;
  }

  setImageFile(image);
};

const onSubmit = async (e) => {
  e.preventDefault();
  let publicUrl = "";

  if (imageFile) {
    publicUrl = await uploadImage();

    axios({
      method: "POST",
      url: "https://detect.roboflow.com/acne_detection/1",

      params: {
        api_key: "G0DIx8ioXilhNZtj1Q4T",
        image: publicUrl,
        confidence: 30,
        format: "JPEG",
      },
    })
      .then(function (response) {
        console.dir(response);
        // const blob = new Blob([response.data], { type: "image/jpeg" });
        // const url = URL.createObjectURL(blob);

        setData(response);
        setImageUrl(publicUrl);
        console.log(data);
      })
      .catch(function (error) {
        console.log(error.message);
      });
  }

  router.replace(router.asPath);
};

return (
  <>
    <Head>
      <title>Create Next App</title>
      <meta name="description" content="Generated by create next app" />
      <meta name="viewport" content="width=device-width, initial-scale=1" />
      <link rel="icon" href="/favicon.ico" />
    </Head>
    <header className={styles.header}>
      <Image src={img}></Image>
      <div>
        {" "}
        <h1>Дипломна робота студента групи КН-42</h1>
        <h1>Комарніцького Іллі</h1>
      </div>
    </header>
    <div className={styles.main}>
      <div className={styles.wrapper}>
        {data && (
```

```
            <ImageWithLabels
              imageWidth={data.data.image.width}
              imageHeight={data.data.image.height}
              labels_arr={data.data.predictions}
              src={imageUrl}
            />
          )}
          {data && <Predictionmap predictions={data.data.predictions} />}
        </div>

        <form
          onSubmit={(e) => onSubmit(e)}
          className={styles.addCategory__form}
        >
          <label htmlFor="image">Зображення категорії</label>
          <input
            accept="image/*"
            type="file"
            name="image"
            id="image"
            ref={imageFileRef}
            onChange={handleImageChange}
          />

          <button type="submit"> submit</button>
        </form>
      </div>
    </>
  );
}

export async function getServerSideProps({ params }) {
  // const supabaseClient = createBrowserSupabaseClient()
  const temp = [];
  const { data: list, error: err } = await supabase.storage
    .from("acne")
    .list("imgs");
  list.map((image) => {
    const { data } = supabase.storage
      .from("acne")
      .getPublicUrl(`imgs/${image.name}`);
    temp.push(data.publicUrl);
  });
  if (err) return alert("no products");
  const temp_url = temp[0];
  return {
    props: { acne: temp },
  };
  // try {
  //   const response = await axios.post(
  //     "http://localhost:3000/api/acne",
  //     temp_url
  //   );
  //   return {
  //     props: { acne: response },
  //   };
  // } catch (error) {
  //   return {
  //     props: {
  //       error: error,
  //     },
  //   };
```

```
  // }
}
import React from "react";

function Predictionmap(predictions) {
  return (
    <div>
      {predictions.predictions.map((item, index) => (
        <p>
          {index + 1} class: {item.class}, x:{item.x}, y:{item.y} , confidence:
          {item.confidence.toFixed(2)}
        </p>
      ))}
    </div>
  );
}

export default Predictionmap;
import { useRef, useEffect, useState } from "react";

function drawRect(ctx, x, y, w, h) {
  ctx.beginPath();
  ctx.rect(x, y, w, h);
  ctx.strokeStyle = "#02a16c";
  ctx.lineWidth = 2;
  ctx.stroke();
}
function drawText(ctx, x, y, text, index) {
  ctx.font = "14px Arial";
  ctx.fillStyle = "#000000";
  index = "N" + index + ": " + text;
  ctx.fillText(index, x, y);
}

export default function ImageWithLabels({
  src,
  imageWidth,
  imageHeight,
  labels_arr,

  //   labelX,
  //   labelY,
  //   labelWidth,
  //   labelHeight,
}) {
  const canvasRef = useRef(null);
  const [widthLabel, setWidth] = useState("");

  useEffect(() => {
    const canvas = canvasRef.current;
    const ctx = canvas.getContext("2d");

    const image = new Image();
    image.onload = () => {
      canvas.width = imageWidth;
      canvas.height = imageHeight;
      const size = imageWidth;

      console.log(size / 100);
      ctx.drawImage(image, 0, 0);

      // draw labels
```

```
      let index = 1;
      for (const item_arr of labels_arr) {
        let labelX = item_arr.x - item_arr.width / 2;
        let labelY = item_arr.y - item_arr.height / 2;
        let labelWidth = item_arr.width;
        let labelHeight = item_arr.height;
        setWidth(item_arr.width);

        drawRect(ctx, labelX, labelY, labelWidth, labelHeight);
        drawText(
          ctx,
          labelX,
          labelY - 5,
          item_arr.confidence.toFixed(2),
          index
        );
        index++;
      }
    };
    image.src = src;
  }, [src, imageWidth, imageHeight, widthLabel]);

  return <canvas ref={canvasRef} />;
}
```