

Міністерство освіти і науки України  
**Національний університет «Запорізька політехніка»**

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

## **Пояснювальна записка**

до дипломного проекту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ  
МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ  
ЕМОЦІЙ ЗА ЗОБРАЖЕННЯМ  
RESEARCH AND SOFTWARE IMPLEMENTATION OF A  
MOBILE APPLICATION FOR IMAGE EMOTION RECOGNITION

Виконав(ла): студент(ка) 2 курсу, групи КНТ-132м  
Спеціальності 121 Інженерія програмного  
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ДЕМ'ЯНОВА Д.К.

(ПРИЗВИЩЕ та ініціали)

Керівник ФЕДОРОНЧАК Т.В.

(ПРИЗВИЩЕ та ініціали)

Рецензент ТЕРЕЩЕНКО Е.В.

(ПРИЗВИЩЕ та ініціали)

Міністерство освіти і науки України  
Національний університет «Запорізька політехніка»

Факультет КНТ  
Кафедра програмних засобів  
Ступінь вищої освіти магістр  
Спеціальність 121 Інженерія програмного забезпечення  
(код і найменування)  
Освітня програма (спеціалізація) Інженерія програмного забезпечення  
(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ПЗ, д.т.н, проф.  
Сергій СУББОТІН  
“ ” 2023 року

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

ДЕМ'ЯНОВОЇ Дар'ї Костянтинівни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація мобільного застосунку для розпізнавання емоцій за зображенням

Research and Software Implementation of a Mobile Application for Image Emotion Recognition

керівник проєкту (роботи) к.т.н., доцент, ФЕДОРОНЧАК Тетяна Василівна,  
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 14 ” листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження.

2. Матеріали і методи. 3. Розробка архітектури програми. 4. Розробка програми.

5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

## 6. Консультанти розділів проєкту (роботи)

| Розділ              | ПРИЗВИЩЕ, ініціали та посада консультанта | Підпис, дата   |                           |
|---------------------|-------------------------------------------|----------------|---------------------------|
|                     |                                           | завдання видав | прийняв виконане завдання |
| 1-5 Основна частина | ФЕДОРОНЧАК Т.В., доцент                   |                |                           |
| Нормоконтролер      | ДЕЙНЕГА Л.Ю., ст. викладач                |                |                           |

7. Дата видачі завдання “ 08 ” вересня 2023 року.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проєкту (роботи)                                  | Строк виконання етапів проєкту ( роботи ) | Примітка     |
|-------|---------------------------------------------------------------------------|-------------------------------------------|--------------|
| 1     | Постановка завдання роботи.                                               | 1 тиждень                                 | Завдання, ТЗ |
| 2     | Аналіз предметної області.                                                | 2-3 тижні                                 | Розділ 1     |
| 3     | Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі. | 4-5 тижні                                 | Розділ 2     |
| 4     | Розробка архітектури програми.                                            | 6 тиждень                                 | Розділ 3     |
| 5     | Розробка програми.                                                        | 7-8 тижні                                 | Розділ 4     |
| 6     | Тестування та експериментальне дослідження програмного забезпечення.      | 9 тиждень                                 | Розділ 5     |
| 7     | Оформлення пояснювальної записки та документів до неї.                    | 10-11 тижні                               | Додатки      |
| 8     | Нормоконтроль та рецензування.                                            | 12 тиждень                                |              |
| 9     | Захист роботи.                                                            | 12 тиждень                                |              |

Студент(ка)

\_\_\_\_\_  
( підпис )      Дар'я ДЕМ'ЯНОВА  
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

\_\_\_\_\_  
( підпис )      Тетяна ФЕДОРОНЧАК  
(Ім'я ПРИЗВИЩЕ)

## РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 106 с., 9 табл., 24 рис., 3 дод., 26 джерел.

FLUTTER, DART, MOBILE, CNN, NEURAL NETWORK, PYTHON, ЕМОЦІЇ, РОЗПІЗНАВАННЯ, BLOC.

Об'єкт дослідження – допомога в розпізнаванні емоцій для людей з розладами аутистичного спектру.

Предмет дослідження – мобільний додаток для визначення та розпізнавання емоцій у людей з розладами аутистичного спектру.

Мета роботи – створення зручного та ефективного інструменту для розпізнавання емоцій, спрямованого на полегшення соціальної взаємодії та покращення якості життя людей з розладами аутистичного спектру.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування Dart, фреймворк Flutter, мова програмування Python, дані для тренування моделі, персональний комп'ютер з процесором Apple M1 під управлінням операційної системи MacOS.

Результати: розроблено та реалізовано мобільний додаток "EmotionCatcher" для платформи Flutter, проведено тренування нейронної мережі з використанням зібраних даних.

Висновки. Розроблений "EmotionCatcher" виступає як ефективний інструмент для розпізнавання емоцій, спрямований на полегшення соціальної адаптації осіб з розладами аутистичного спектру.

Галузь використання – допомога в розпізнаванні емоцій людей.

## ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 106 pages, 24 figures, 9 tables, 3 appendixes, 26 sources.

FLUTTER, DART, MOBILE, CNN, NEURAL NETWORK, PYTHON, EMOTIONS, RECOGNITION, BLOC.

Object of research: assistance in recognizing emotions for individuals with autism spectrum disorders.

Subject of research: the mobile application for identifying and recognizing emotions in individuals with autism spectrum disorders.

Objective: creating a convenient and effective tool for emotion recognition, aimed at facilitating social interaction and improving the quality of life for individuals with autism spectrum disorders.

Materials, methods, and technical means: object-oriented programming, Dart programming language, Flutter framework, Python programming language, data for model training, a personal computer with an Apple M1 processor running MacOS.

Results: developed and implemented the "EmotionCatcher" mobile application for the Flutter platform, trained a neural network using collected data.

Conclusions: the developed "EmotionCatcher" serves as an effective tool for emotion recognition, aiming to facilitate social adaptation for individuals with autism spectrum disorders.

The field of use is assistance in recognizing emotions in people

## ЗМІСТ

|                                                               |    |
|---------------------------------------------------------------|----|
|                                                               | С. |
| Перелік скорочень та умовних познач. ....                     | 6  |
| Вступ.....                                                    | 9  |
| 1 Аналіз предметної області .....                             | 10 |
| 1.1 Опис проблеми.....                                        | 10 |
| 1.2 Задача класифікації .....                                 | 11 |
| 1.3 Типи класифікації .....                                   | 13 |
| 1.4 Класифікація зображень .....                              | 13 |
| 1.5 Алгоритми глибокого навчання .....                        | 17 |
| 1.6 Огляд існуючих аналогів .....                             | 23 |
| 1.7 Постановка завдання до роботи .....                       | 26 |
| 2 Матеріали і методи .....                                    | 27 |
| 2.1 Підготовка даних .....                                    | 27 |
| 2.2 Побудова Моделі CNN .....                                 | 27 |
| 2.3 Функція втрат перехресної ентропії .....                  | 28 |
| 2.4 Оптимізатор .....                                         | 33 |
| 2.5 Висновки за розділом 2 .....                              | 34 |
| 3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ .....                         | 35 |
| 3.1 Вимоги до застосунку .....                                | 35 |
| 3.2 Вибір інструментів розробки програми .....                | 36 |
| 3.3 Вибір бази даних .....                                    | 45 |
| 3.4 Архітектура застосунку .....                              | 45 |
| 3.5 Компоненти системи .....                                  | 46 |
| 3.6 Висновки за розділом 3 .....                              | 48 |
| 4 РОЗРОБКА ПРОГРАМИ .....                                     | 49 |
| 4.1 Основні рішення щодо розробки бекенду .....               | 49 |
| 4.2 Основні рішення щодо розробки мобільного застосунку ..... | 52 |
| 4.3 Структура бази даних .....                                | 55 |
| 4.4 Алгоритми програми .....                                  | 56 |

|     |                                                                        |    |
|-----|------------------------------------------------------------------------|----|
| 4.4 | Опис розроблених інтерфейсів .....                                     | 57 |
| 4.5 | Висновки за розділом 4 .....                                           | 62 |
| 5   | Експлуатація, тестування та експрементальне дослідження програми ..... | 63 |
| 5.1 | Призначення та умови застосування програми .....                       | 63 |
| 5.2 | Виконання програми .....                                               | 63 |
| 5.3 | Тестування програми .....                                              | 68 |
| 5.4 | Висновки за розділом 5 .....                                           | 71 |
|     | Висновки .....                                                         | 72 |
|     | Перелік джерел посилання .....                                         | 74 |
|     | Додаток А Технічне завдання .....                                      | 77 |
|     | Додаток Б Текст програми .....                                         | 81 |
|     | Додаток В Слайди презентації .....                                     | 98 |

**ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК**

|      |                                   |
|------|-----------------------------------|
| API  | Application Programming Interface |
| CNN  | Convolutional Neural Network      |
| DJL  | Deep Java Library                 |
| DOM  | Document Object Model             |
| FER  | Facial Emotion Recognition        |
| JVM  | Java Virtual Machine              |
| LSTM | Write Once, Run Anywhere          |
| ML   | Machine Learning                  |
| MLP  | Multilayer Perceptron             |
| ReLU | Rectified Linear Unit             |
| RNN  | Recurrent Neural Network          |
| WORA | Write Once, Run Anywhere          |
| ПЗ   | Програмне забезпечення            |
| РАС  | Розлади аутистичного спектру      |

## ВСТУП

У сучасному світі, де стрімкі технологічні зміни переплітаються зі складними соціокультурними викликами, розуміння та взаємодія з різними аспектами людського життя набуває нових розмірів. Однією з важливих сфер цієї взаємодії є розуміння та виявлення емоцій, зокрема серед тих, хто має розлади аутистичного спектру (РАС). Допомога особам з РАС у виявленні та розумінні емоцій є високоактуальною проблемою, що вимагає інноваційних рішень. Особливості сприйняття та вираження емоцій у цій групі людей роблять це завдання складним та важливим для покращення їхнього повсякденного життя та соціальної адаптації.

Розробка та використання нейронних мереж стає важливим інструментом для автоматизованого розпізнавання емоцій, особливо в контексті РАС. Нейронні мережі дозволяють створювати моделі, які можуть виявляти нюанси емоційних виразів та допомагати з визначенням зображеної емоції.

Метою даної дипломної роботи є розробка програмного забезпечення (ПЗ) розпізнавання емоцій, спрямованого на підтримку осіб з РАС. Спрощення виявлення емоцій через використання передових технологій, таких як нейронні мережі, може значно полегшити їхню взаємодію у різних соціальних сценаріях.

Існуючі мобільні застосунки не в повній мірі задовольняють потреби цільової аудиторії. Тому було вирішено створити програмний продукт, який забезпечить функціоналом розпізнавання емоцій на зображеннях та створення корисної для користувача статистики.

Ця робота прагне внести вагомий внесок у поле досліджень, спрямованих на покращення якості життя осіб з РАС та підтримку їхньої соціальної інтеграції.

## 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Розпізнавання емоцій на зображеннях може бути сформульовано як задача класифікації, оскільки передбачення емоцій пов'язано з присвоєнням конкретного класу (емоції) кожному зображенню. У цьому контексті класифікація використовується для визначення того, до якого класу належить об'єкт, що розпізнається – емоційний стан на даному зображенні.

### 1.1 Опис проблеми

Люди з розладами аутистичного спектру (РАС) мають труднощі з соціально-емоційним функціонуванням; однак дослідження розпізнавання емоцій на обличчі (FER) залишаються безрезультатними. Люди з РАС можуть використовувати нетипові компенсаторні механізми, які виснажуються під час більш складних завдань. У одному з досліджень порівнювали точність і швидкість відповіді на завдання FER із примусовим вибором, використовуючи нейтральні вирази, вирази радості, смутку, відрази, гніву, страху та здивування як у хронометражних, так і в вільних умовах у дітей із та без РАС ( $n = 18$ ). Результати показали, що точність розпізнавання емоцій була порівнянна в обох групах в умовах без часу. Однак, у хронометражних умовах діти з РАС були менш точними у визначенні гніву та здивування порівняно з дітьми без РАС. Це свідчить про те, що люди з РАС мають нетипову обробку гніву та здивування, які можуть бути викликані під тиском часу. Розуміння цих нетипових процесів і факторів навколишнього середовища, які їм загрожують, може бути корисним для підтримки соціально-емоційного функціонування людей з РАС [1].

Виявлено, що людям із захворюваннями аутичного спектру бракує навичок емоційного інтелекту. Ці недоліки пов'язані з порушенням емоційного усвідомлення, вираження та регуляції та порушенням здатності до емпатичного розуміння та поведінки. Крім того, ці порушення

спостерігаються в оцінках, що перевіряють виявлення емоцій за виразом обличчя, словесною інтонацією та мовою тіла, а також у екологічних, життєвих завданнях, що вимагають інтеграції емоційних сигналів з різних каналів сприйняття в контексті. Незважаючи на те, що деякі люди з РАС встановлюють методи компенсації, які дозволяють їм виявляти фундаментальні емоційні вираження та обставини, загальна слабкість у розпізнаванні більш складних емоцій зберігається в дорослому віці. Декілька досліджень показали, що ці програми можуть допомогти дітям з РАС сприймати та виражати свої емоції [1].

Створення застосунку для розпізнавання емоцій стане одним з механізмів для дорослих і дітей з РАС, щоб поліпшити навички власного розпізнавання емоцій в навколишньому світі. Для розпізнавання і виявлення емоції на фото потрібен підхід машинного навчання і створення власної нейронної мережі.

## **1.2 Задача класифікації**

Машинне навчання зазвичай поділяють на два основних типи. У прогнозованому або наглядному підході до навчання мета полягає в тому, щоб навчитися відображати входи  $x$  на виходи  $y$ , враховуючи позначені набори пар входів-виходів. Оскільки навчання передбачає взаємодію між учнем і навколишнім середовищем, можна розділити навчальні завдання відповідно до характеру цієї взаємодії. Перша відмінність, на яку слід звернути увагу - це різниця між навчанням під наглядом і без нього.

Більш абстрактно, розглядаючи навчання як процес «використання досвіду», навчання під наглядом описує сценарій, у якому «досвід», приклад навчання, містить важливу інформацію, якої немає в невидимих «тестових прикладах» до якого має бути застосований отриманий досвід. У цьому налаштуванні отриманий досвід спрямований на прогнозування інформації, якої бракує для тесту даних. У таких випадках ми можемо думати про

середовище як про вчителя, який «контролює» учня, надаючи додаткову інформацію в неконтрольоване навчання, проте немає різниці між навчанням і тестовими даними. Учень обробляє вхідні дані з метою прийти з деяким підсумком або стисненою версією цих даних [2].

У цьому розділі ми обговоримо класифікацію. Мета полягає в тому, щоб вивчити відображення вхідних даних  $x$  на виходи  $y$ , де  $y \in \{1, \dots, C\}$ , де  $C$  є кількістю класів. Якщо  $C = 2$ , це називається бінарною класифікацією (у цьому випадку ми часто припускаємо, що  $y \in \{0, 1\}$ ); якщо  $C > 2$ , це називається багатокласова класифікація. Якщо мітки класів не є взаємовиключними (наприклад, хтось може класифікуватися як високий і сильний), ми називаємо це класифікацією з кількома мітками, але це найкраще розглядати як прогнозування кількох пов'язаних двійкових міток класу (так звана модель множинного виведення). Коли ми використовуємо термін «класифікація», ми маємо на увазі багатокласову класифікацію з одним виходом, якщо тільки ми стверджуємо інше.

Один із способів формалізувати проблему - це апроксимація функції. Ми припускаємо  $y = f(x)$  для деякої невідомої функції  $f$ , і метою навчання є оцінка функції  $f$ , заданої позначеною навчальним набором, а потім прогноз за допомогою  $\hat{y} = \hat{f}(x)$  (позначення оцінки) Головна мета — робити прогнози щодо нових вхідних даних, тобто тих, які ми не бачили раніше (це називається узагальненням), з моменту прогнозування відповіді на навчальному наборі [3].

Глибоке навчання використовує штучні нейронні мережі, які імітують структуру людського мозку. Штучна нейронна мережа складається з кількох прихованих шарів, які містять штучні нейрони. Під час аналізу зображень модель глибокого навчання витягує характеристики з необроблених даних і пропускає їх через взаємопов'язані вузли. Моделі глибокого навчання навчаються так само, як і базові моделі машинного навчання. Вони вчаться пов'язувати зображення з мітками за допомогою тренувань під наглядом, без або напів під наглядом. Що відрізняє модель глибокого навчання, так це

здатність витягувати, аналізувати та розуміти складні зв'язки та представлення даних зображень у набагато більшій величині.

Незважаючи на те, що моделі глибокого навчання є набагато потужнішими, вони мають більше параметрів, гірше інтерпретуються та займають більше часу для навчання. Це підкреслює важливість ефективного процесу анотації, який впливає на час навчання та продуктивність моделі.

### **1.3 Типи класифікації**

Бінарна класифікація застосовує логічне порівняння до об'єкта та класифікує невідомі точки даних за однією з двох категорій. Наприклад, система візуального контролю на виробничому підприємстві класифікує продукти як браковані або готові до продажу після аналізу знімків у реальному часі. Тим часом класифікація за кількома мітками дозволяє програмному забезпеченню додавати декілька класів до одного об'єкту. Наприклад, системи обробки зображень, які живлять автономні транспортні засоби, позначають пішоходів, автомобілі, дороги та будівлі на тих самих візуальних даних [4].

### **1.4 Класифікація зображень**

Розпізнавання емоцій на зображеннях виражається через класифікаційну задачу, де присвоєння конкретного класу (емоції) кожному зображенню дозволяє передбачити емоційний стан об'єкта розпізнавання. У даному контексті класифікація служить засобом визначення, до якого класу відноситься емоційний вислів на зазначеному зображенні.

В особливому випадку, коли зображення складаються з окремих рукописних літер і цифр, наприклад, у поштовій адресі чи поштовому індексі на листі ми можемо використовувати класифікацію для визначення рукописного введення.

Багато загальних методів класифікації ігнорують будь-яку структуру вхідних ознак, наприклад просторовий макет. Отже, вони також можуть так само легко обробляти дані, які виглядають як на рис 1.1.

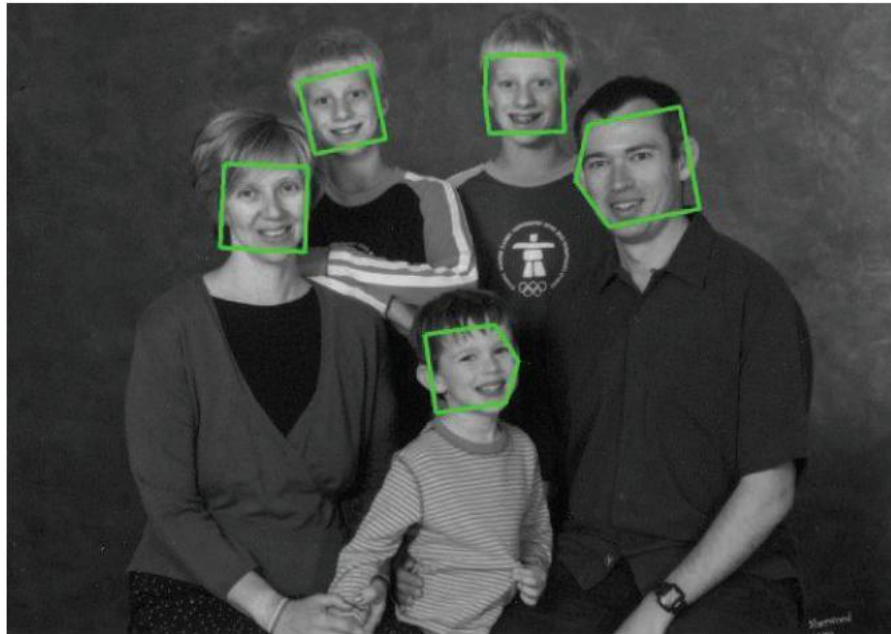


Рисунок 1.1 - Приклад розпізнавання обличчя. Вихід класифікатора, який виявив 5 облич в різних позах [3]

Проблема більш складна - це знаходження об'єктів на зображенні; це називається виявленням об'єктів чи локалізацією об'єктів. Важливий випадок – це виявлення обличчя. Один з підходів до цієї задачі полягає в поділі зображення на багато невеликих перекриваючихся патчів у різних місцях, масштабах і орієнтаціях. Після цього проводиться класифікація кожного патчу з використанням аналізу текстури, подібної до обличчя, або відмінної від нього. Це називається детектором ковзного вікна. Система потім повертає ті місця, де ймовірність обличчя достатньо висока.

Знайшовши обличчя, можна продовжити розпізнавання облич, що означає визначення особи. У цьому випадку кількість класів може бути дуже великою. Також ознаки, які слід використовувати, ймовірно, будуть відмінні від проблеми виявлення обличчя: для розпізнавання можуть бути важливі

невеликі відмінності між обличчями, такі як зачіска, але для виявлення важливо бути інваріантним до таких деталей і просто фокусуватися на відмінностях між обличчями та не-обличчями [3].

Розпізнавання облич та емоцій є ключовими завданнями в сучасних системах інтелектуального аналізу емоцій, які мають широкий спектр застосувань від реклами та маркетингу до медицини та безпеки. Крім того, розвиток комп'ютерного зору та машинного навчання робить ідентифікацію емоцій набагато точнішою та доступною для населення. У результаті розпізнавання виразу обличчя швидко розвивається як підполе обробки зображень. Деякі з можливих застосувань – це взаємодія людини з комп'ютером, психіатричні спостереження, розпізнавання п'яних водіїв, детектор брехні [5]. У зв'язку з цим, подальший розвиток та дослідження методів розпізнавання емоцій зображеннях є актуальним та перспективним напрямком в сучасній інформатиці та комп'ютерній науці.

Метою процесу класифікації є віднесення всіх пікселів цифрового зображення до одного з кількох класів. Зазвичай для виконання класифікації використовуються мультиспектральні дані, і, справді, спектральна картина, наявна в даних для кожного пікселя, використовується як числова основа для категоризації. Мета класифікації зображень полягає в тому, щоб ідентифікувати та відобразити, як унікальний рівень сірого (або колір), особливості, які зустрічаються на зображенні з точки зору об'єкта, який ці особливості фактично представляють на землі. Класифікація зображень є, мабуть, найважливішою частиною аналізу цифрових зображень. Класифікація об'єктів є складним завданням, тому класифікація зображень була важливою задачею в області комп'ютерного зору. Класифікація зображень відноситься до позначення зображень в один із кількох заздалегідь визначених класів. Потенційно існує  $n$  класів, до яких можна класифікувати дане зображення. Ручна перевірка та класифікація зображень може бути виснажливим завданням, особливо коли їх величезна кількість, тому буде дуже корисно, якщо ми зможемо автоматизувати весь цей процес за

допомогою комп'ютерного зору. Досягнення в галузі автономного водіння також є чудовим прикладом використання класифікації зображень у реальному світі. Програми включають автоматизовану організацію зображень, вебсайти зі стоковими фотографіями та відео, візуальний пошук для кращої видимості продукту, великі візуальні бази даних, розпізнавання зображень і облич у соціальних мережах та багато іншого. Ось чому нам потрібні класифікатори для досягнення максимально можливої точності [6].

Процес класифікації зображень зазвичай включає п'ять кроків.

1. Вибір та підготовка зображень. В залежності від того, що має бути класифіковано, слід вибрати найбільш відповідний сенсор, найбільш відповідні дати отримання та найбільш відповідні спектральні діапазони.

2. Визначення кластерів у просторі ознак. Тут можливі два підходи: навчана та ненавчана класифікація. У навчаній класифікації оператор визначає кластери під час процесу навчання; у ненавчаній класифікації алгоритм кластеризації автоматично знаходить та визначає кількість кластерів у просторі ознак.

3. Вибір алгоритму класифікації. Після визначення спектральних класів у просторі ознак оператор повинен вирішити, як пікселі (на основі їхніх векторів ознак) будуть призначатися класам. Призначення може базуватися на різних критеріях.

4. Проведення фактичної класифікації. Після визначення тренувальних даних та вибору алгоритму класифікації можна провести фактичну класифікацію. Це означає, що кожен "багатосмуговий піксель" (комірка) на зображенні призначається одному з попередньо визначених класів.

5. Перевірка результату. Після отримання класифікованого зображення його якість оцінюється порівнянням з даними-посиланням. Це вимагає вибору техніки вибірки, створення матриці помилок та розрахунку параметрів помилок [7].

## 1.5 Алгоритми глибокого навчання

У цьому розділі проводиться глибокий аналіз та порівняння різних методів обробки зображень, які використовуються для розпізнавання облич та виявлення емоцій. Ці методи є видами Deep Learning. Deep Learning - це застосування штучних нейронних мереж із використанням сучасного обладнання. Це дозволяє розробляти, навчати та використовувати нейронні мережі, які є набагато більшими, ніж вважалося можливим раніше. Для розробки варто зосередитись на наступних видах нейронних мереж:

- Multilayer Perceptrons (MLPs);
- Convolutional Neural Networks (CNNs);
- Recurrent Neural Networks (RNNs).

В цьому пункті буде розглянуто кожен з цих видів нейронних мереж, вивчаючи їхню структуру, переваги та обмеження в контексті розпізнавання облич та емоцій. Такий аналіз допоможе краще зрозуміти, які методи обробки зображень найбільш підходять для конкретних завдань та даних.

Також можливе розширення інформації про кожен тип нейронної мережі, включаючи приклади їх застосувань та конкретні викиди використання в різних галузях.

### 1.5.1 Multilayer Perceptrons

Multilayer Perceptrons, або скорочено MLP, є класичним типом нейронної мережі. Вона складається з одного або кількох шарів нейронів. Дані подаються на вхідний рівень, може існувати один або кілька прихованих шарів, що забезпечують рівні абстракції, а передбачення робляться на вихідному рівні, який також називають видимим шаром.

MLP підходять для задач прогнозування класифікації, де входам призначається клас або мітка. Вони також підходять для задач регресійного прогнозування, де прогнозується дійсна величина за набором вхідних даних.

Дані часто надаються в табличному форматі, наприклад, у файлі CSV або електронній таблиці.

MLP доречно використовувати для табличних наборів даних, класифікаційних задачі прогнозування і задач регресійного прогнозування. Вони є дуже гнучкими і ця гнучкість дозволяє застосовувати їх до інших типів даних [8].

Ця нейронна мережа також може розрізняти дані, які не є лінійно роздільними. Недоліком є те, що загальна кількість параметрів може зрости до дуже високої (кількість персептронів у шарі 1, помножена на  $p$  у шарі 2, помножена на  $p$  у шарі 3...). Це неефективно, оскільки в таких високих розмірах є надмірність. Іншим недоліком є те, що вона ігнорує просторову інформацію і сприймає сплюснені вектори як вхідні дані [9].

### 1.5.2 Convolutional Neural Network

Convolutional Neural Network (згорткові найромережі) відрізняються від інших нейронних мереж своєю чудовою продуктивністю з зображенням, мовленням або аудіосигналом. Вони мають три основні типи шарів, а саме:

- згортковий шар;
- шар об'єднання;
- повністю підключений рівень.

Convolutional layer (або згортковий шар) є першим шаром найромережі. У той час як за згортковими шарами можуть слідувати додаткові згорткові шари або шари об'єднання, повністю зв'язаний рівень є останнім. З кожним шаром CNN зростає у своїй складності, ідентифікуючи більші частини зображення. Попередні шари зосереджені на простих функціях, таких як кольори та краї. Коли дані зображення просуваються між шарами CNN, вони починають розпізнавати більші елементи або форми об'єкта, поки нарешті не ідентифікують запланований об'єкт. Після кожної

операції згортки CNN застосовує перетворення Rectified Linear Unit (ReLU) до карти ознак, вносячи нелінійність у модель (рис. 1.2).

Об'єднання шарів, також відоме як зменшення дискретизації, виконує зменшення розмірності, зменшуючи кількість параметрів у вхідних даних. Подібно до згорткового шару, операція об'єднання очищає фільтр по всьому входу, але відмінність полягає в тому, що цей фільтр не має вагових коефіцієнтів. Замість цього ядро застосовує функцію агрегації до значень у приймальному полі, заповнюючи вихідний масив. Існує два основних типи об'єднання.

1. Максимальне об'єднання: під час руху фільтра по входу він вибирає піксель із максимальним значенням для надсилання у вихідний масив. Крім того, цей підхід зазвичай використовується частіше, ніж середнє об'єднання.

2. Об'єднання середніх значень: під час руху фільтра по входу він обчислює середнє значення в приймальному полі для надсилання у вихідний масив.

Хоча багато інформації втрачається на рівні об'єднання, це також має ряд переваг для CNN. Вони допомагають зменшити складність, підвищити ефективність і обмежити ризик переобладнання [10].

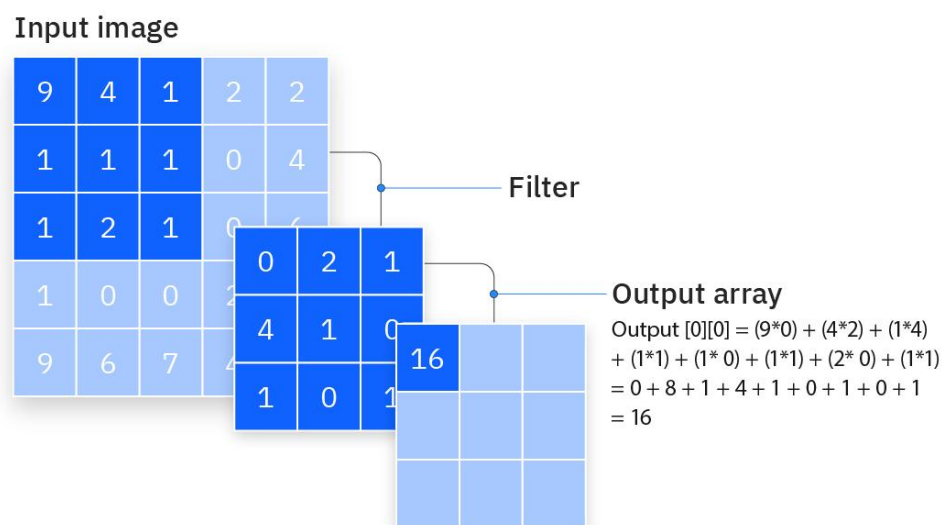


Рисунок 1.2 - Перетворення ReLU [104]

### 1.5.3 Recurrent Neural Networks

Мережі з повторювальними зв'язками, або RNN, розроблені для роботи з задачами прогнозування послідовностей. Такі задачі можуть приймати різні форми, залежно від типів введення та виведення.

Приклади задач прогнозування послідовностей включають:

- один-до-багатьох: спостереження на вході, яке відображається в послідовність з кількома кроками на виході;
- багато-до-одного: послідовність кількох кроків на вході, яка відображається як прогноз класу чи кількості на виході;
- багато-до-багатьох: послідовність кількох кроків на вході, яка відображається в послідовність з кількома кроками на виході, що часто називається "sequence-to-sequence" або "seq2seq".

Мережі Long Short-Term Memory (LSTM) є одними з найуспішніших типів RNN, оскільки вони подолали проблеми тренування рекурентних мереж і використовуються в широкому спектрі застосувань. RNN, загалом, та LSTM, зокрема, найкраще працюють з послідовностями слів і абзаців, так званою обробкою природної мови. Вони також використовуються в якості генеративних моделей для послідовностей, наприклад, генерації рукопису.

Використовується RNN для:

- текстових даних;
- даних з мовлення;
- проблем класифікації;
- проблем регресії;
- генеративних моделей.

RNN не підходять для табличних наборів даних у форматі CSV або електронних таблиць і не є відповідними для обробки зображень.

### 1.5.4 Порівняння нейронних мереж

Разом з тим, для отримання більш глибокого розуміння та визначення найкращого підходу до розпізнавання облич та емоцій на зображеннях, ми складаємо порівняльну таблицю (табл. 1.1), яка допоможе систематизувати і порівняти основні характеристики різних типів нейронних мереж: Multilayer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), і Recurrent Neural Networks (RNNs).

Таблиця 1.1 - Порівняння видів нейронних мереж глибокого навчання

| Критерій                                            | MLP | CNN | RNN |
|-----------------------------------------------------|-----|-----|-----|
| Здатність до розпізнавання облич та виразів обличчя | +   | ++  | +   |
| Обробка зображень                                   | -   | ++  | -   |
| Відмінності у текстурі та грануляції                | -   | ++  | -   |
| Здатність до аналізу послідовних даних              | -   | -   | +   |
| Табличні набори даних                               | +   | -   | -   |
| Класифікаційні задачі прогнозування                 | +   | +   | +   |
| Задачі регресійного прогнозування                   | +   | +   | +   |
| Швидкість навчання                                  | -   | ++  | ++  |
| Кількість параметрів                                | ++  | ++  | ++  |
| Точність розпізнавання                              | -   | +   | +   |
| Обсяг навчальних даних                              | -   | +   | +   |
| Застосування для аналізу послідовних даних (відео)  | -   | +   | +   |

Ця порівняльна таблиця надає загальний огляд та порівняння трьох основних типів нейронних мереж щодо їхніх здатностей до розпізнавання облич та емоцій на зображеннях.

Очевидно, що кожен тип мережі має свої унікальні переваги та обмеження. MLPs підходять для табличних даних та класифікаційних задач, тоді як CNNs ідеально підходять для обробки зображень та RNNs для послідовних даних. Важливо вибирати тип мережі відповідно до конкретного завдання та характеристик даних.

Для задач розпізнавання емоцій на зображеннях, часто найбільш оптимальним та ефективним вибором є використання згорткових нейронних мереж (CNN). Цей вибір обумовлений декількома ключовими факторами.

CNN були спеціально розроблені для роботи з зображеннями, що робить їх ідеальними для завдань розпізнавання облич та емоцій. Вони автоматично вивчають різноманітні ознаки, такі як кольори, текстури, форми, а також відносини між пікселями у зображеннях. Також вони демонструють вражаючу точність у розпізнаванні виразів обличчя та емоцій, оскільки вони вивчають унікальні ознаки, пов'язані з виразами на обличчях, такі як розташування очей, брови, рота та інші ключові деталі. Згорткові нейронні мережі добре справляються з обробкою великих обсягів зображень та навчанням на великих наборах даних, що є важливим для точного розпізнавання емоцій.

Отже, на основі вищезазначених факторів та порівняння з іншими типами нейронних мереж, для завдань розпізнавання емоцій на зображеннях, очевидним є вибір CNNs для вирішення поставленої задачі. Використання CNNs допомагає досягти високої точності та ефективності у завданнях розпізнавання облич та виразів обличчя.'

## **1.6 Огляд існуючих аналогів**

### **1.6.1 FaceAnalyzer :Age, Gender**

FaceAnalyzer: Age, Gender – мобільний застосунок, розміщений у AppStore і доступний для встановлення на iOS платформі. Це простий застосунок який дозволяє по зображенню розпізнати емоцію, вік та гендер особи [11].

FaceAnalyzer: Age, Gender - додаток для аналізу віку та емоцій на зображеннях облич. Основні функції включають:

- аналіз віку, здатність визначати приблизний вік особи на фотографії за допомогою обробки зображення;
- розпізнавання емоцій, автоматичне визначення емоцій на обличчі та подання результату у вигляді графіка;
- графік емоцій, перегляд графічного представлення емоцій, що дозволяє користувачам більш детально розуміти їхні емоційні стани.

FaceAnalyzer вирізняється завдяки таким особливостям:

- графічний аналіз емоцій: Надає користувачам можливість переглядати емоційний стан у формі графіку, що робить додаток більш привабливим та зрозумілим;
- простота використання, інтуїтивний та простий інтерфейс для швидкого та зручного використання додатку;
- можливість обрати джерело зображення, камера чи галерея девайсу.

FaceAnalyzer: Age, Gender надає користувачам розширені функції та інтерактивні можливості для вивчення емоцій та вікових характеристик на зображеннях облич. На рис. 1.3 показані основна структура, дизайн застосунку та функціонал.

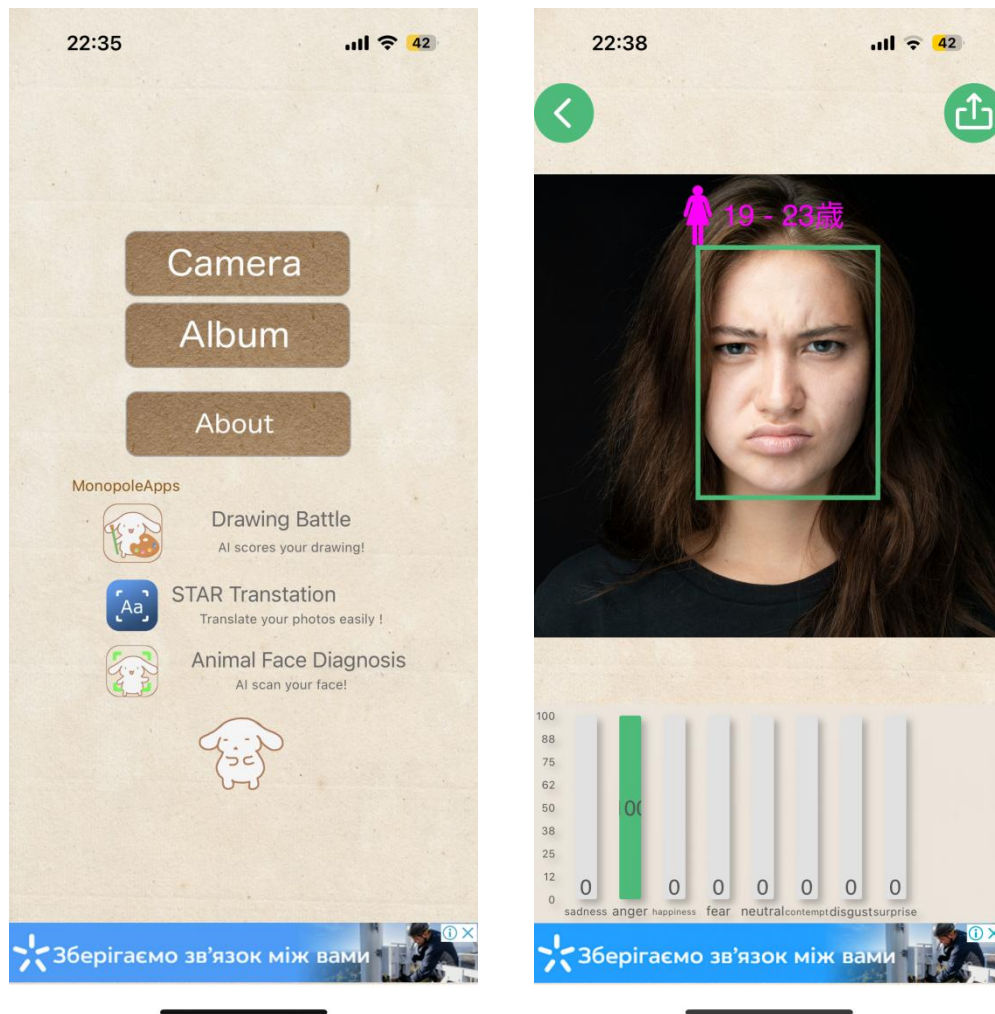


Рисунок 1.3 – Інтерфейс застосунку FaceAnalyzer

### 1.6.2 Emotimetr

Emotimeter – це мобільний застосунок, розміщений у Play Market і доступний для завантаження на Android пристрої. Цей застосунок відрізняється достатньою популярністю на відміну від попереднього аналогу [12]. На рис. 1.4 показаний інтерфейс застосунку та функціонал щодо розпізнавання емоцій на зображеннях та відео.

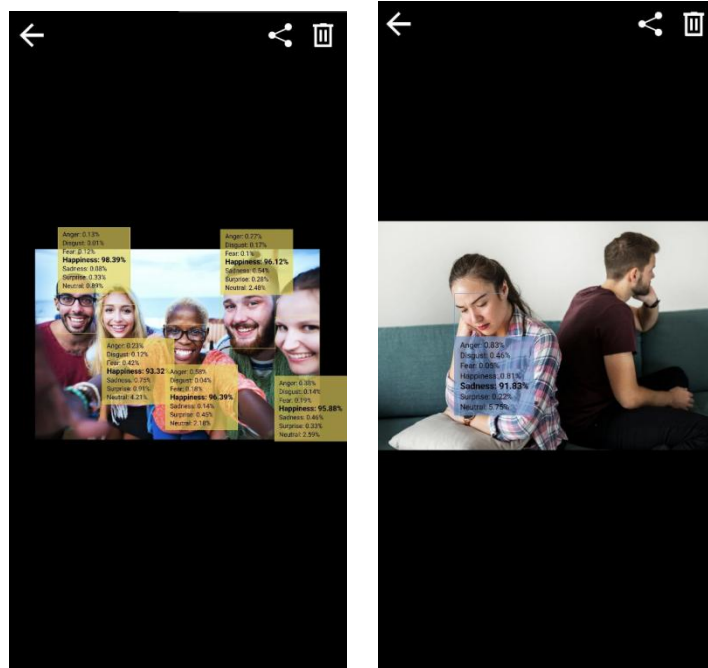


Рисунок 1.4 – Інтерфейс застосунку Emotimeter [12]

Основні функції Emotimeter:

- виявлення емоцій в режимі реального часу, здатність аналізувати емоції на основі міміки з обличчя в реальному часі;
- створення фотографій і відео в реальному часі, здатність фіксувати моменти виявлення емоцій у вигляді фотографій або відео;
- можливість аналізувати емоції на зображеннях та відео, збережених у галереї користувача;
- можливість виявлення емоцій у кожному кадрі відео.

Emotimeter визначається своєрідністю реального часу та можливістю аналізу емоцій як в режимі живого взаємодії, так і при перегляді збережених матеріалів, надаючи користувачам різні можливості використання.

### 1.6.3 Порівняння аналогів

Аналоги застосунків було проаналізовано і відокремлено основні переваги й недоліки (табл. 1.2). Наявність останніх вказує на недосконалість

продукту для кінцевого користувача або на те, що розглянута проблема не може бути повною мірою вирішена за допомогою наведених рішень.

Таблиця 1.2 – Порівняння існуючих аналогів

| Критерій                                                             | FaceAnalyzer | Emotimeter |
|----------------------------------------------------------------------|--------------|------------|
| Кросплатформеність                                                   | —            | —          |
| Обробка зображень з різних джерел                                    | +            | +          |
| Збереження історії фото                                              | —            | +          |
| Популярність застосунку                                              | —            | +          |
| Можливість користувача “вгадати” емоцію перед розпізнаванням моделлю | —            | —          |
| Багатозадачність                                                     | +            | —          |

Отже, проаналізувавши аналоги, можна визначити які проблеми мають бути вирішені в ПЗ, що розробляється.

### 1.7 Постановка завдання до роботи

За результатами дослідження проблеми було сформовано головне завдання - створення мобільного застосунку, який за допомогою нейронної мережі CNN буде розпізнавати емоції в зображеннях. Також було сформовано технічне завдання, яке представлено в додатку А.

## **2 МАТЕРІАЛИ І МЕТОДИ**

### **2.1 Підготовка даних**

У цьому розділі розглянуто процес підготовки даних для тренування та тестування моделі розпізнавання емоцій на зображеннях. Використано набір даних, який містить зображення різних емоцій, розподілені за класами.

Використовуються зображення розміром 48x48 пікселів зі збереженням кольорової інформації. Процес підготовки включає читання та зміну розміру зображень, а також застосування методів аугментації даних, таких як градація сірого, відображення, розмиття та інші.

Для завдання розпізнавання емоцій обрано архітектуру згорткової нейронної мережі (CNN). Модель складається з кількох шарів згортки та повністю з'єднаних шарів, а також використовує методи регуляризації для уникнення перенавчання.

### **2.2 Побудова Моделі CNN**

В даному випадку, для розпізнавання емоцій, використовується згорткова нейронна мережа (CNN) з такою архітектурою.

1. Розмір зображень: (ширина, висота, канали) - (48, 48, 1).
2. Згорткові Шари (Convolutional Layers):
  - перший згортковий шар: 32 фільтри, розмір ядра (3, 3), активація ReLU, обнулення (padding), стрід (strides) (1, 1);
  - другий згортковий шар: 64 фільтри, розмір ядра (3, 3), активація ReLU, обнулення, стрід (1, 1);
  - третій згортковий шар: 128 фільтрів, розмір ядра (3, 3), активація ReLU, обнулення, стрід (1, 1);
  - четвертий згортковий шар: 256 фільтрів, розмір ядра (3, 3), активація ReLU, обнулення, стрід (1, 1).

3. Шари Пулінгу (Pooling Layers): MaxPooling2D з розміром пулінгу (2, 2).
4. Повністю З'єднані Шари (Fully Connected Layers): 128 нейронів, активація ReLU.
5. Вихідний Шар (Output Layer). Повністю з'єднаний шар з 3 нейронами (кількість класів для класифікації емоцій).
6. Компіляція Моделі. Використання функції втрат - Cross Entropy Loss. Використання метрика для оцінки продуктивності.

Ця архітектура використовується для тренування моделі на тренувальних даних і розпізнавання емоцій на нових зображеннях. Такий підхід може бути ефективним для задач класифікації емоцій на основі зображень.

### 2.3 Функція втрат перехресної ентропії

Ентропія - це середня кількість бітів, необхідних для представлення або передачі події, взятої з розподілу ймовірностей для випадкової змінної. Простими словами, ентропія вказує на ступінь невизначеності події.

У галузі науки про дані перехресна ентропія між двома дискретними розподілами ймовірностей пов'язана з Розходженням Кульбака — Лейблера, метрикою, яка фіксує, наскільки схожі два розподіли. Враховуючи справжній розподіл  $t$  і прогнозований розподіл  $p$ , перехресна ентропія між ними визначається такою формулою (2.1):

$$H(t, p) = - \sum_{s \in S} t(s) \cdot \log(p(s)) \quad (2.1)$$

Перехресна ентропія розширює концепцію ентропії теорії інформації шляхом вимірювання варіації між двома розподілами ймовірностей для даної випадкової змінної/набору випадків. Вона використовується під час коригування ваг моделі під час навчання. Мета полягає в тому, щоб

мінімізувати втрати — чим менші втрати, тим краща модель. Ідеальна модель має перехресну втрату ентропії 0. Зазвичай вона обслуговує класифікації з кількома класами та кількома мітками.

Втрата перехресної ентропії вимірює різницю між виявленим і прогнозованим розподілами ймовірностей моделі класифікації глибокого навчання.

Softmax - це функція, а не втрата. Вона стискає вектор в діапазоні (0, 1), і всі отримані елементи додаються до 1. Застосовується до вихідних балів  $s$ . Оскільки елементи представляють клас, їх можна трактувати як ймовірності класу. Функцію Softmax не можна застосувати незалежно до кожного елемента  $s_i$ , оскільки вона залежить від всіх елементів  $s$ . Для заданого класу  $s_i$  функцію Softmax можна обчислити за формулою (2.2):

$$f(s_i) = \frac{e^{s_i}}{\sum_j^c e^{s_j}} \quad (2.2)$$

Функції активації перетворюють вектори перед обчисленням втрати на етапі навчання. Під час тестування функції активації також використовуються для отримання виходів CNN, коли втрата вже не застосовується.

Cross-entropy (крос-ентропія) розширює концепцію ентропії теорії інформації, вимірюючи варіацію між двома розподілами ймовірностей для заданої випадкової величини/набору випадків. Вона використовується при налаштуванні ваг моделі під час тренування з метою мінімізації втрат - чим менше втрат, тим краща модель. Ідеальна модель має втрату крос-ентропії 0. Вона зазвичай служить для багатокласової та багатоетикеткової класифікації.

Крос-ентропія вимірює відмінність між виявленим та передбаченим розподілами ймовірностей моделі глибокого навчання. Між двома розподілами ймовірностей, такими як Q від P, вона формально визначається

як  $H(P, Q)$ , де  $H()$  - це функція крос-ентропії,  $P$  - це цільовий розподіл,  $Q$  - наближення цільового розподілу.

Крос-ентропію можна обчислити, використовуючи ймовірності подій від  $P$  та  $Q$  за формулою (2.3):

$$H(P, Q) = - \sum_{x \in X} P(x) \cdot \log(Q(x)) \quad (2.3)$$

Зазвичай перед обчисленням втрати  $CE$  застосовується функція активації (Sigmoid/Softmax). За допомогою Softmax модель передбачає вектор ймовірностей  $[0.7, 0.2, 0.1]$ . Сума 70%, 20% і 10% дорівнює 100%, і перший елемент є найбільш ймовірним. Softmax перетворює логіти в ймовірності (рис. 2.1)[13].

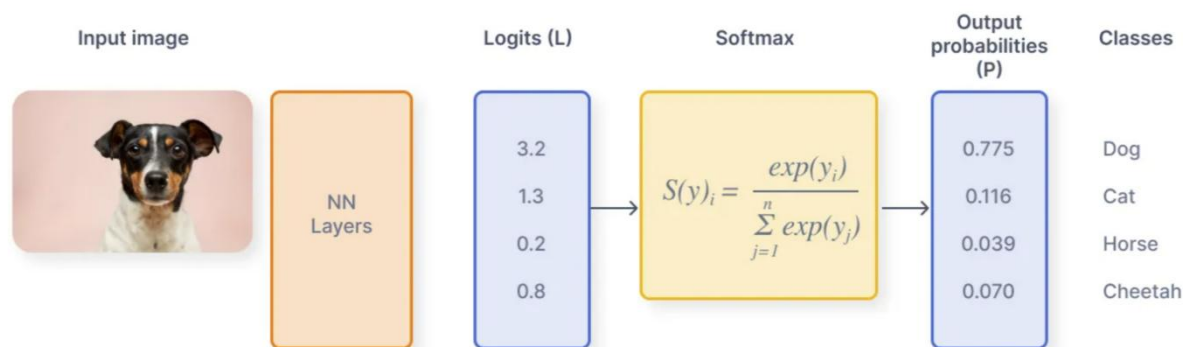


Рисунок 2.1 — Класифікація зображення з використанням крос-ентропійних втрат ( $S$  — Softmax вихід,  $T$  — ціль) [13]

Мета крос-ентропії полягає в тому, щоб взяти вихідні ймовірності ( $P$ ) і виміряти відстань від істинних значень як на рис. 2.2.

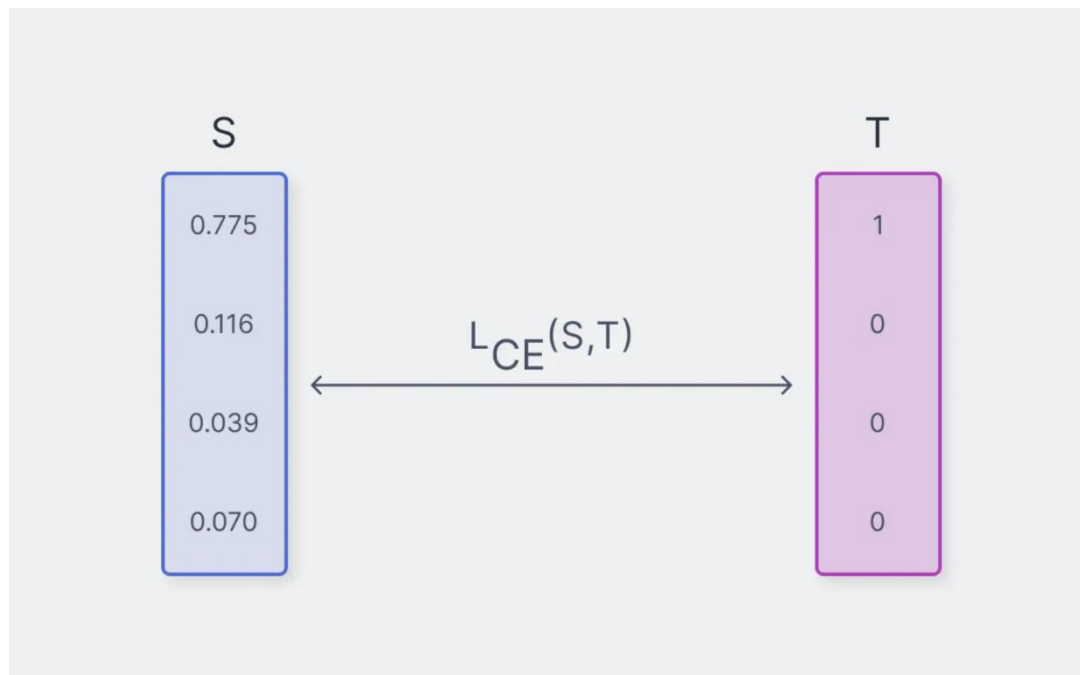


Рисунок 2.2 — Крос ентропія (L), S — Softmax вихід, T — ціль [14]

На рис. 2.3 показано вхідний параметр функції втрат крос-ентропії.

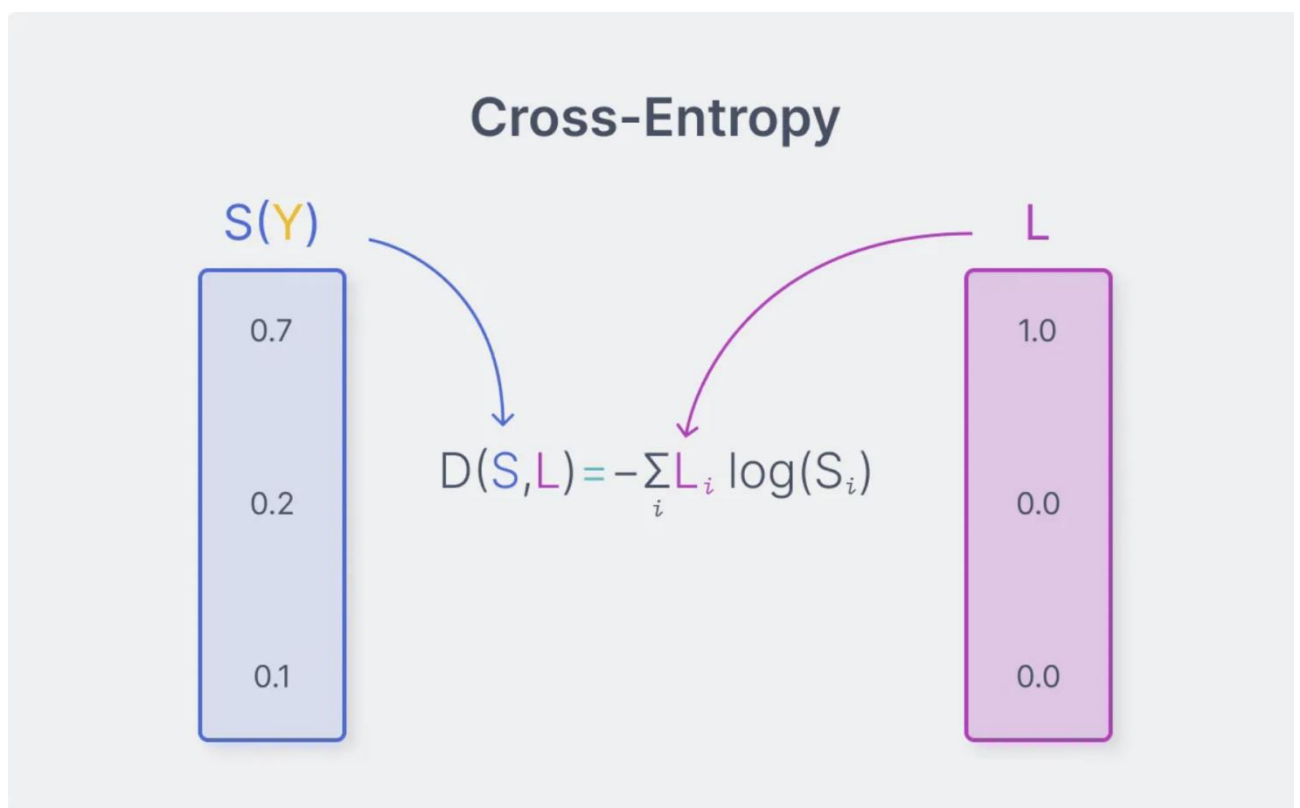


Рисунок 2.3 — Вхідні параметри функції втрат крос-ентропії [14]

Перехресна втрата ентропії визначається за формулою (2.4) як:

$$CE = - \sum_i^C t_i \log(s_i) \quad (2.4)$$

де  $t_i$  і  $s_i$  це визначена істина та оцінка CNN для кожного класу  $i$  в  $C$ .

Оскільки зазвичай функція активації (Sigmoid / Softmax) застосовується до балів перед обчисленням втрати  $CE$ , ми пишемо  $f(s_i)$  посилаючись на активацію.

У задачі двійкової класифікації, де  $C'=2$ , перехресну втрату ентропії можна також визначити за формулою (2.5) як:

$$CE = - \sum_{i=1}^{C'=2} t_i \log(s_i) = - t_1 \log(s_1) - (1 - t_1) \log(1 - s_1), \quad (2.5)$$

де передбачається, що є два класи:  $C_1$  і  $C_2$ .  $t_1 \in [0,1]$  і  $s_1$  є визначеною істиною та оцінкою для  $C_1$ , і  $t_2 = 1 - t_1$ ,  $s_2 = 1 - s_1$  є визначеною істиною та оцінкою для  $C_2$ .

Це той випадок, коли ми розділяємо проблему класифікації Multi-Label в  $C$  задачі бінарної класифікації. Логістичні втрати та мультиноміальні логістичні втрати — інші назви перехресних ентропійних втрат.

Категоричні крос-ентропійні втрати також називаються Softmax Loss. Це активація Softmax плюс втрата крос-ентропії. Якщо ми використовуємо цю втрату, ми навчимо CNN виводити ймовірність  $C$  класів для кожного зображення. Цей підхід можна використовувати для багатокласової класифікації [15].

Використання функції втрат Cross Entropy Loss у програмі для розпізнавання емоцій обрано через природу завдання класифікації, яке включає визначення емоцій на зображеннях. Оскільки у завданні існують різні класи емоцій, Cross Entropy Loss дозволяє ефективно оптимізувати ймовірності приналежності кожного зразка до визначених класів. Цей метод

є стандартним вибором для багатокласових класифікаційних задач, і він сприяє покращенню точності класифікації в множині емоційних станів.

## 2.4 Оптимізатор

Оптимізатор у задачах класифікації використовується для налаштування параметрів моделі з метою мінімізації втрати під час тренування. Оптимізатор шукає оптимальні значення ваг моделі, щоб підвищити точність передбачень. Оптимізатори використовують різні алгоритми для ефективного оновлення ваг моделі під час тренування. Популярні оптимізатори, такі як "Adam", "SGD" (Stochastic Gradient Descent), "RMSprop" і т.д., мають різні підходи до цього процесу та параметри, які можуть бути налаштовані. Adam - це алгоритм оптимізації, широко використовуваний у глибокому навчанні. Це розширення алгоритму стохастичного градієнтного спуску SGD і призначене для оновлення ваг нейронної мережі під час тренування. Adam адаптивно налаштовує швидкість навчання для кожної окремої ваги, динамічно обчислюючи її на основі минулих градієнтів та їхніх других моментів. У відмінність від SGD, який утримує постійну швидкість навчання, Adam адаптивно обчислює індивідуальні швидкості навчання[16].

Цей оптимізатор поєднує корисні властивості інших алгоритмів оптимізації, таких як AdaGrad і RMSProp. Відмінність полягає в тому, що Adam обчислює невцентровану дисперсію градієнтів (без віднімання середнього), аналогічно RMSProp, який віднімає середнє.

Загалом, Adam - це алгоритм оптимізації, який розширює SGD, динамічно адаптуючи швидкості навчання для кожної окремої ваги. Його ефективність полягає в тому, що він комбінує властивості AdaGrad і RMSProp, що забезпечує ефективні та адаптивні оновлення ваг під час тренування глибокої нейронної мережі.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \left[ \frac{\delta L}{\delta \omega_t} \right] \quad v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left[ \frac{\delta L}{\delta \omega_t} \right]^2 \quad (2.6)$$

Наведена формула (2.6) представляє роботу оптимізатора Адама. Тут  $\beta_1$  і  $\beta_2$  представляють швидкість спаду середнього значення градієнтів.

Однак Adam має свої недоліки, і вибір алгоритму оптимізації залежить від конкретних вимог і характеру даних [16].

## 2.5 Висновки за розділом 2

Вибір функції втрат перехресної ентропії та оптимізатора Adam є обґрунтованим для задачі класифікації зображень з розпізнаванням емоцій. Функція втрат перехресної ентропії є відповідною для багатокласових задач, де модель повинна призначати кожному класу ймовірності. Оптимізатор Adam дозволяє адаптивно регулювати швидкість навчання для кожної ваги, що полегшує оптимізацію та пришвидшує збіжність моделі. Це особливо важливо в глибокому навчанні, де швидкість навчання може значно впливати на ефективність тренування. Такий вибір допомагає досягти кращої продуктивності та точності моделі в завданні розпізнавання емоцій на зображеннях.

### 3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

#### 3.1 Вимоги до застосунку

В результаті дипломної роботи має бути розроблений мобільний клієнт-серверний застосунок для розпізнавання емоцій. Функціональні вимоги до застосунку можна описати наступним чином:

- фотографування; користувач повинен мати можливість зробити фотографію за допомогою мобільного застосунку;
- відправка зображення на бекенд; зображення, зроблене користувачем, повинно відправлятися на бекенд для подальшого аналізу;
- розпізнавання емоції; бекенд повинен використовувати розроблену модель для розпізнавання емоцій на зображенні;
- отримання результату; мобільний застосунок повинен отримати результат розпізнавання емоції з бекенда;
- порівняння результатів; користувач повинен мати можливість порівняти результат розпізнавання емоції, отриманий від моделі, із своїм власним сприйняттям емоції на зображенні;
- збереження історії; результати розпізнавання та порівняння повинні зберігатися в історії для подальшого перегляду;
- перегляд історії; користувач повинен мати можливість переглядати попередні результати розпізнавання емоцій збережених зображень.

Ці функціональні вимоги визначають основні можливості застосунку та взаємодію між мобільним клієнтом та бекендом.

Також наведено нефункціональні вимоги.

1. Продуктивність. Бекенд повинен надавати результат розпізнавання відправленого зображення протягом максимально допустимого часу (10 с), щоб користувач не чекав надто довго і дані швидко відображалися.

2. Надійність. Бекенд повинен бути стійким до навантаження та відновлювати роботу після можливих збоїв. Має бути забезпечена стабільна

робота мобільного застосунку під час різних умов зв'язку та можливих помилок.

3. Сумісність і підтримка різних платформ (Android, iOS) для розширення аудиторії користувачів.

4. Інтуїтивний користувацький інтерфейс. Інтерфейс мобільного застосунку має бути простим та інтуїтивно зрозумілим для користувачів будь-якого рівня технічної підготовки.

5. Масштабованість. Бекенд повинен бути здатний масштабуватись для обробки збільшеного навантаження в разі зростання кількості користувачів.

7. Споживання ресурсів. Ефективне використання ресурсів мобільного пристрою (батареї, пам'яті) для зручності користувачів.

8. Автентифікація та авторизація. Створення механізмів автентифікації та авторизації для збереження конфіденційності та безпеки даних користувачів.

Ці нефункціональні вимоги допоможуть забезпечити ефективну та надійну роботу застосунку для розпізнавання емоцій.

## **3.2 Вибір інструментів розробки програми**

### **3.2.1 Вибір мови програмування для бекенду і нейронної мережі**

#### **3.2.1.1 Java**

Java — це високорівнева об'єктно-орієнтована мова програмування на основі класів, розроблена таким чином, щоб мати якомога менше залежностей реалізації. Це мова програмування загального призначення, призначена для того, щоб програмісти могли написати один раз і запустити будь-де – Write Once, Run Anywhere(WORA), тобто скомпільований код Java може працювати на всіх платформах, які підтримують Java, без необхідності повторної компіляції. Програми Java зазвичай скомпільовані в байт-код, який може працювати на будь-якій віртуальній машині Java, наприклад Java Virtual

Machine (JVM), незалежно від базової архітектури комп'ютера. Синтаксис Java подібний до C і C++, але має менше можливостей низького рівня, ніж обидва з них. Середовище виконання Java надає динамічні можливості (такі як відображення та модифікація коду середовища виконання), які зазвичай недоступні в традиційних скомпільованих мовах[17].

Java протягом тривалого часу була однією з найпопулярніших мов програмування на підприємствах і має величезну екосистему бібліотек і фреймворків, а також велику спільноту розробників. Однак у Java є дуже обмежені варіанти програм глибокого навчання. Зараз більшість моделей глибокого навчання написані та навчаються на Python. Це створює додаткову перешкоду для розробників Java, які хочуть увійти в цю сферу, оскільки їм потрібно вивчити як нову мову програмування, так і складну область глибокого навчання.

Щоб знизити бар'єр для входу в глибоке навчання для Java-розробників, AWS створив Deep Java Library (DJL), фреймворк глибокого навчання. А також існують Apache MXNet, PyTorch або TensorFlow для запуску навчання та логічного висновку на Java. Він також містить потужний дизайн ModelZoo, який дозволяє вам керувати навченими моделями та завантажувати їх в одному рядку коду. Вбудований ModelZoo наразі підтримує понад 70 попередньо навчених і готових до використання моделей від GluonCV, HuggingFace, TorchHub і Keras[18].

Java визначається великою популярністю та екосистемою, які забезпечують стабільність та широкий спектр інструментів для розробників. Фреймворки, такі як Spring та Apache Struts, сприяють швидкій та масштабованій розробці бекенд-рішень. Java відома своєю надійністю та здатністю ефективно обробляти великі обсяги даних, роблячи її відмінним вибором для створення надійних та масштабованих бекендів.

Хоча Python традиційно вважається основною мовою для глибокого навчання, Java надає можливості для створення та впровадження моделей глибокого навчання. TensorFlow, одна з провідних бібліотек для роботи з

глибоким навчанням, надає підтримку для Java, дозволяючи розробникам використовувати мову для навчання та використання моделей CNN у різних доменних областях, включаючи обробку зображень та комп'ютерний зір.

### 3.2.1.2 JavaScript

JavaScript, JS, — це мова програмування, яка є однією з основних технологій Всесвітньої павутини, поряд з HTML і CSS. Станом на 2023 рік 98,7% вебсайтів використовують JavaScript на стороні клієнта для поведінки вебсторінок, часто використовуючи сторонні бібліотеки. Усі основні веббраузери мають спеціальний механізм JavaScript для виконання коду на пристроях користувачів.

JavaScript — це мова високого рівня, яка часто скомпільована точно вчасно і відповідає стандарту ECMAScript. Він має динамічний тип, об'єктну орієнтацію на основі прототипу та першокласні функції. Це мультипарадигма, яка підтримує керований подіями, функціональний та імперативний стилі програмування. Він має інтерфейси прикладного програмування (API) для роботи з текстом, датами, регулярними виразами, стандартними структурами даних і об'єктною моделлю документа (DOM) [19]. TensorFlow.js — це нова версія популярної бібліотеки з відкритим кодом, яка забезпечує глибоке навчання JavaScript. Тепер розробники можуть визначати, навчати та запускати моделі машинного навчання за допомогою бібліотечного Application Programming Interface (API) високого рівня. Попередньо навчені моделі означають, що розробники тепер можуть легко виконувати такі складні завдання, як візуальне розпізнавання, створення музики чи визначення людських поз лише за допомогою кількох рядків JavaScript. Розпочавшись як інтерфейсна бібліотека для веббраузерів, останні оновлення додали експериментальну підтримку Node.js. Це дозволяє використовувати TensorFlow.js у внутрішніх програмах JavaScript без використання Python[20].

### 3.2.1.3 Python

Python — це мова програмування високого рівня загального призначення. Його філософія дизайну наголошує на зручності читання коду з використанням значних відступів.

Python динамічно типізується та збирає сміття. Він підтримує кілька парадигм програмування, включаючи структуроване (зокрема процедурне), об'єктно-орієнтоване та функціональне програмування [21]. Python — це об'єктно-орієнтована мова програмування загального призначення з відкритим вихідним кодом, яку хвалять за високу читабельність, продуктивність і швидкість. Python часто використовується як основна мова серверної частини або мова розширення для вебзастосунків, написаних іншими мовами, а також його часто вибирають, коли йдеться про розробку платформ автоматизації, інтелектуального аналізу даних і машинного навчання. Python є однією з офіційних серверних мов, що використовуються Google; він також використовується в інфраструктурах Instagram і Facebook. Крім того, Quora, Netflix і Spotify широко використовують його, головним чином для забезпечення своїх можливостей аналізу даних і серверних служб [22]. Розробка штучного інтелекту та машинного навчання швидко прогресує, і в бізнесі не бракує інноваційних застосувань. Деякі яскраві приклади включають класифікатори зображень, аналітику настроїв у соціальних мережах, чат-боти, інтелектуальні механізми або персоналізовані рекомендації. Щоб реалізувати ці інструменти та функції, дослідники даних, програмісти та DevOps використовують комбінацію мов програмування, включаючи Python.

Протягом багатьох років Python був мовою вибору для реалізацій Machine Learning (ML). Він надає повну бібліотеку пакетів із вбудованими функціями, які полегшують аналіз і обробку даних, очищення, моделювання, візуалізацію тощо. До них відносяться TensorFlow, Keras і Theano, які спрощують реалізацію різних функцій машинного навчання. Багато

розробників вважають Python кращою мовою для проектів ML також через його здатність взаємодіяти з іншими мовами та платформами, а також надійну здатність обробки даних. Порівняння мов програмування було наведено в табл. 3.1.

Таблиця 3.1 – Порівняння мов програмування

| Критерій              | Java                                                                   | JavaScript                                                                                                                           | Python                                                                                    |
|-----------------------|------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| Призначення           | Бекенд-розробка, корпоративні застосунки                               | Бекенд-розробка (Node.js), фронтенд, мобільний фронтенд (React Native)                                                               | Загальні програми, веброзробка, глибоке навчання                                          |
| Популярність          | Висока                                                                 | Висока                                                                                                                               | Дуже висока                                                                               |
| Екосистема            | Велика, широкий вибір фреймворків та бібліотек                         | Велика, прм для пакетів Node.js                                                                                                      | Велика, використання фреймворків (Django, Flask), багато бібліотек для глибокого навчання |
| Швидкість роботи      | Висока                                                                 | Висока                                                                                                                               | Висока                                                                                    |
| Асинхронність         | Зазвичай синхронна, але є асинхронні фреймворки (Spring WebFlux)       | Асинхронна (зокрема у Node.js)                                                                                                       | Зазвичай синхронна, але є асинхронні бібліотеки та фреймворки                             |
| Глибоке навчання      | Так, існують бібліотеки (TensorFlow, Deeplearning4j)                   | Є TensorFlow.js для глибокого навчання                                                                                               | Дуже популярна мова для глибокого навчання, багато бібліотек (TensorFlow, PyTorch, Keras) |
| Бекенд-розробка       | Підходить для розробки надійних та масштабованих серверних застосунків | Популярна у веброзробці за допомогою Node.js та Express.js, швидка та легко масштабована, особливо для невеликих і середніх проектів | Використовується для веброзробки та серверної розробки.                                   |
| Зручність в навчанні  | Середня                                                                | Легка                                                                                                                                | Легка                                                                                     |
| Підтримка фреймворків | Spring, Hibernate, Apache Struts                                       | Node.js, Nest.js,                                                                                                                    | Django, Flask, FastAPI                                                                    |

Ця таблиця надає загальний порівняльний огляд Java, JavaScript та Python з огляду на різні критерії. Враховуючи розглядані критерії та порівняння між Java, JavaScript та Python для бекенд-розробки та нейронних

мереж, обрано Python. Python відзначається великою популярністю, широкою екосистемою, легкістю вивчення, а також високою підтримкою для розробки бекенду (фреймворки Django та Flask) та глибокого навчання (бібліотеки TensorFlow, PyTorch, Keras). Ця мова надає ефективні інструменти для створення надійних серверних додатків та реалізації складних моделей нейронних мереж, забезпечуючи гнучкість та продуктивність в обох областях.

### **3.2.2 Вибір мови програмування мобільного застосунку**

Flutter — це платформа інтерфейсу користувача з відкритим кодом, розроблена Google, яка дозволяє розробникам створювати мобільні, вебпрограми та програми для настільних комп'ютерів нативної якості з єдиної кодової бази. Flutter — це цінний сучасний інструмент, який використовується для створення приголомшливих кросплатформних програм, які відтворюють рідний код на кожному пристрої та в ОС. Flutter сумісний з Android, iOS, Linux, macOS, Windows тощо. Flutter — це багатоплатформена технологія, яка максимізує обмін кодами та забезпечує взаємодію з різними платформами. Розробник створює оригінальний код, який стає ексклюзивним для кожної платформи, замість створення рідних рішень. Flutter допомагає компаніям економити їхні бюджети на розробку, оскільки немає потреби наймати рідних розробників додатків, вам просто потрібен професійний інженер Flutter, який створить функціональні можливості, а інструмент зробить все інше. Він сумісний з мобільними платформами (iOS і Android), настільними платформами (Windows, Linux, MacOS), а також браузерами. У результаті підхід спільного коду використовується для швидкої розробки програмного забезпечення будь-якої складності та робить Flutter одним із найвідоміших інструментів для швидкої та економічно ефективної розробки MVP.

Flutter має наступні переваги [23].

1. Однаковий код для всіх платформ. Flutter дозволяє розробникам створювати один код для різних платформ, таких як веб, настільні та мобільні пристрої, що призводить до швидшого запуску додатка та економії витрат.
2. Зменшений час розробки. Flutter дозволяє створювати більші додатки за менший час, що сприяє ефективній розробці та знижує витрати на обслуговування.
3. Швидкість введення на ринок. Flutter має високу реактивність, що дозволяє швидко виводити продукт на ринок, особливо в умовах невизначеності термінів для iOS та Android версій.
4. Нативна продуктивність. Використання графічного движка Skia дозволяє Flutter створювати швидкі та оптимізовані додатки, нерозрізненні від нативних за рахунок відсутності посередників чи інтерпретаторів коду. Flutter забезпечує вбудовану підтримку для Material Design (Android) та Cupertino (iOS), забезпечуючи нативний вигляд на обох платформах та створюючи консистентний та знайомий користувацький досвід.
5. Власний рендеринговий двигун. Flutter має унікальні пакети для обох операційних систем, уникнення схожості між iOS та Android.
6. Функція гарячого перевантаження. Можливість швидкої перезавантаження є однією з ключових переваг Flutter, що прискорює розробку додатків.
7. Безпека Flutter. Null-safety на рівні синтаксису та ізольовані пісочниці для обчислювальних потоків гарантують відсутність небезпечних ресурсів.
8. Мова Dart. Програмування на Dart є ключовою причиною вибору Flutter. Застосування AOT- та JIT-компіляторів робить Flutter гнучким та ефективним для розробки.
9. Відмінна документація та підтримка спільноти. Flutter має широку документацію та активну спільноту, що робить навчання та вирішення проблем більш простими.

Окрім переваг, звісно, наявні недоліки Flutter як фреймворку.

1. Великі та важкі додатки: Необхідно враховувати недолік у розмірі розроблюваних додатків у Flutter. Розробники програм, які працюють з цим інструментарієм, можуть зіткнутися з труднощами у роботі з великими файлами, що може призвести до вибору більш легкого альтернативного рішення.

2. Низька популярність Dart. Dart- надійна мова програмування, але не є популярною, особливо порівняно з іншими провідними мовами програмування, такими як Java, Kotlin і т. д.

3. Проблеми з iOS: Flutter працює добре на обох платформах - Android та iOS. Проте той факт, що Flutter розроблено Google, надає перевагу додаткам для Android.

4. Обмежена кількість сторонніх бібліотек: Flutter, порівняно з мовами програмування на рівні системи, має обмежену кількість сторонніх бібліотек. Розробники все ще повинні витратити більше часу на створення власних бібліотек.

Отже, Flutter був обраний для розробки мобільного застосунку для розпізнавання емоцій бо дозволяє розробку кросплатформених застосунків, що означає, що один і той самий код може використовуватися для створення застосунків як для Android, так і для iOS. Flutter надає гнучкість та швидкість розробки завдяки гарному інструментарію, багатим набором віджетів та ефективній системі гарячої перезавантаження. Крім того, Flutter має активну та зростаючу спільноту розробників, що забезпечує доступ до багатьох готових рішень та підтримку.

### **3.2.3 Вибір середовища розробки**

VS Code отримав масу прихильників у спільноті розробників завдяки своїй легкій природі та великій бібліотеці розширень. Це редактор коду з відкритим вихідним кодом, розроблений корпорацією Майкрософт,

розроблений таким чином, щоб мати широкі можливості налаштування, залишаючись ефективним і швидким. Що стосується розробки Flutter, VS Code пропонує кілька переваг.

VS Code відомий своєю швидкою продуктивністю навіть на менш потужних машинах. Це робить його привабливим варіантом для розробників, які хочуть плавного кодування без шкоди для швидкості. Ринок VS Code може похвалитися величезною кількістю розширень, спеціально розроблених для розробки Flutter. Ці розширення забезпечують такі функції, як IntelliSense, підтримка налагодження та гаряче перезавантаження, покращуючи робочий процес розробки.

Завдяки мінімалістичному дизайну VS Code дозволяє розробникам адаптувати свою IDE відповідно до своїх уподобань. Можна вибрати з безлічі тем, кольірних схем і розширень, щоб створити робочий простір, який відповідає вашому стилю. Також, VS Code має вбудований термінал, який усуває необхідність перемикатися між редактором і окремим вікном терміналу. Ця інтеграція спрощує завдання та зберігає все в одному місці[24].

Працюючи з Python та Flutter у Visual Studio Code, використання розширень Microsoft Python та Flutter, відповідно, виявляється надзвичайно зручним і продуктивним. Розширення для Python надає повний функціонал редактора для розробки з мовою Python, де можна ефективно використовувати автоматичне завершення коду, лінтування, налагодження та інші корисні можливості, працюючи з різними інтерпретаторами. Тим часом, розширення для Flutter робить VS Code ідеальним інструментом для розробки мобільних додатків, забезпечуючи зручний інтерфейс для написання, тестування та налагодження коду з використанням Flutter. Отже, обидва розширення відмінно взаємодіють із середовищем розробки, забезпечуючи продуктивний та комфортний процес створення як Python-проектів, так і мобільних додатків за допомогою Flutter. Окремо можна виділити розширення для роботи з системами контролю версій,

автогенерацією, автоматизацією, тестування запитів в Postman та автозаповнення.

### **3.3 Вибір бази даних**

MongoDB обрано для цього проекту з розпізнавання емоцій з кількох вагомих причин. По-перше, гнучкість сховища даних дозволяє адаптувати схему даних без перерви в роботі, що особливо важливо в проектах, де вимоги до структури даних можуть змінюватися. Друге, MongoDB забезпечує швидкодію завдяки своєму документ-орієнтованому підходу та ефективним операціям зберігання та отримання даних у форматі документів. Третє, база даних легко масштабується, що дозволяє додавати нові сервери для розподілу навантаження та забезпечення масштабованості проекту.

### **3.4 Архітектура застосунку**

Для розробки бекенд частини застосунку була обрана монолітна архітектура. Такий підхід може бути обраний для простоти та зручності у випадках, коли розмір проекту ще невеликий, і немає потреби в складній розділеній структурі.

Переваги монолітної архітектури включають простоту розгортання та управління проектом, особливо на етапі початкового розвитку. Однак, з ростом функціональності та розмірів проекту, може виникнути потреба у більш розділених та масштабованій архітектурі для полегшення розвитку та управління складністю.

Для мобільного застосунку була обраний підхід на основі Clean Architecture. Обрання цієї архітектури може бути обґрунтоване кількома перевагами. Clean Architecture розділяє код на рівні відповідальності (Entities, Use Cases, Interface Adapters, Frameworks & Drivers), що полегшує розуміння та модифікацію коду. Також дозволяє розділити бізнес-логіку від деталей

реалізації, а значить мобільний застосунок може легко адаптуватися до змін у фреймворках або платформах без значного впливу на бізнес-логіку.

Чиста архітектура підтримує тестування, можна писати автономні тести для бізнес-логіки. Код, організований за принципами Clean Architecture, є менш залежним від конкретної технології і більш довговічним, оскільки зміни в технічних деталях не сильно впливають на бізнес-логіку.

Розділення в чистій архітектурі буде відбуватись на 3 шари – Domain, Data, Presentation. Це дозволяє виокремити окремо бізнес-логіку і Use Cases в шар Domain. Також він буде містити обраний State Management для застосунку. Шар Presentation включатиме в себе повністю UI частину застосунку- всі екрани, модальні вікна тощо. З цього шару буде відбуватись відправка запитів або подій до Domain. Presentation не знає про реалізацію бізнес-логіки застосунку, отже може просто використовувати створені сервіси. Data Layer – цей шар відповідає за доступ до даних зовні застосунку, як наприклад, зв'язок з сервером або доступ до локального сховища даних (рис 3.1).

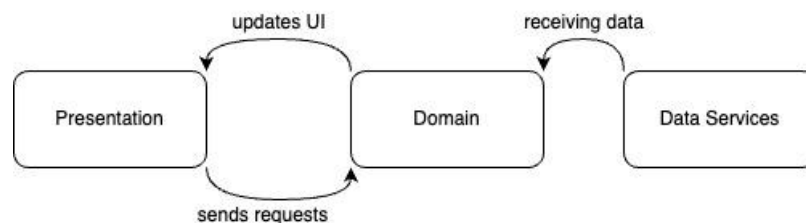


Рисунок 3.1 – Архітектура мобільного застосунку

### 3.5 Компоненти системи

Після визначення архітектури застосунку необхідно створити більш детальну діаграму компонентів системи (рис. 3.2). Вона складається з семи компонентів.

1. Мобільний застосунок (Mobile App). Відповідає за інтерфейс користувача, фотографування та ініціювання розпізнавання емоцій.

2. Бекенд (Backend). Забезпечує API для мобільного застосунку та включає ендпоінти для авторизації, реєстрації користувачів, передачі зображень для розпізнавання та інші.

3. Сервіс розпізнавання емоцій (Emotion Recognition Service). Використовує нейронну мережу для розпізнавання емоцій на основі обробки зображень.

4. Нейронна мережа (Neural Network Model). Містить попередньо навчену модель для розпізнавання емоцій на зображеннях.

5. Сервіс історії (History Service). Зберігає та надає доступ до історії розпізнавань користувача.

6. Модуль авторизації (Authorization). Включає сервіси авторизації та реєстрації користувачів.

7. Middleware для авторизації (Authorization Middleware). Забезпечує перевірку доступу та авторизацію для API ендпоінтів.

Мобільний застосунок інтерактивно взаємодіє з бекендом через API. Бекенд взаємодіє з сервісами розпізнавання емоцій, обробки зображень та іншими. Сервіс розпізнавання емоцій використовує нейронну мережу та оброблює зображення. Сервіс історії обробляє та зберігає історію розпізнавань. Модуль авторизації та middleware для авторизації взаємодіють для обробки запитів авторизації та реєстрації.

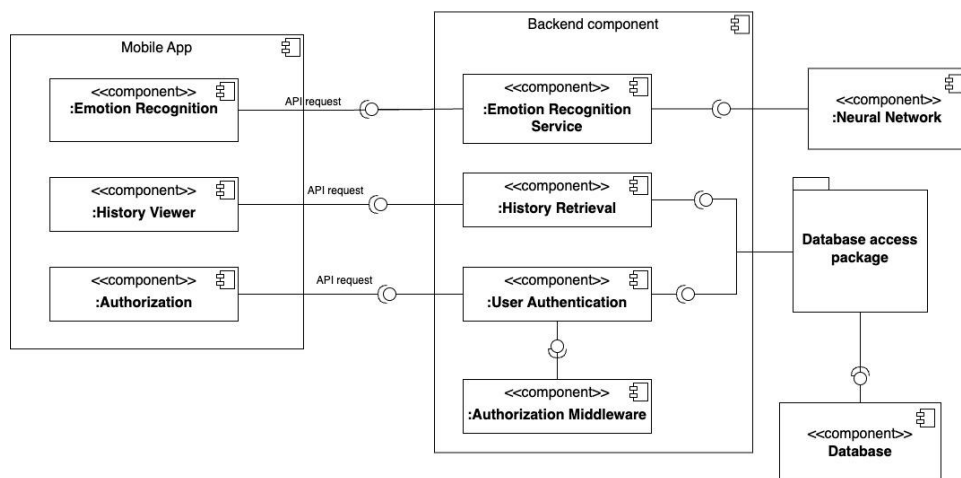


Рисунок 3.2 - Діаграма компонентів

Отже, в застосунку, що буде розроблений, основний функціонал кінцевого користувача буде складатись із двох частин - перегляд історії та власне розпізнавання емоції за фото. Попередньо користувач має визначити, яка на його думку є емоція, зображена на фото. Відповідно до цього рішення користувач отримує оцінку успішності його вибору. Діаграма прецедентів для застосунку зображена на рис.3.3.

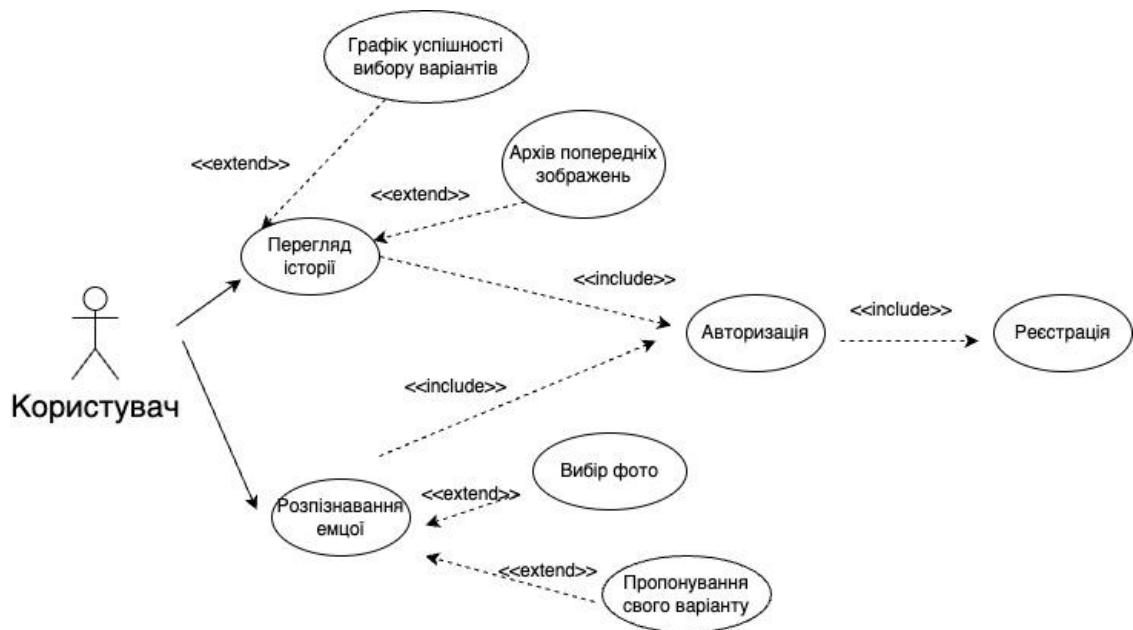


Рисунок 3.3 - Діаграма прецедентів

### 3.6 Висновки за розділом 3

В цьому розділі були порівняні мови програмування для розробки бекенд частини застосунку та нейронної мережі, було обране середовище для розробки ПЗ. Була також визначена мова програмування для створення мобільного застосунку та БД MongoDB. На основні обраної архітектури була створена відповідна схема ПЗ та описані основні компоненти системи, що розроблюється.

## 4 РОЗРОБКА ПРОГРАМИ

### 4.1 Основні рішення щодо розробки бекенду

Ця частина застосунку складається з CNN та бекенд застосунку для комунікації з БД і мобільним застосунком. Основними класами є EmotionModel, EmotionProcessingHelper, DBHelper, UserHandler, EmotionsHandler, HistoryHandler, App.

Вхідними даними є обраний датасет для навчання моделі, а також API запити з мобільного застосунку.

#### 4.1.1 Клас EmotionModel

Основна мета класу EmotionModel - створити нейромережу CNN та зберегти навчену модель для наступного використання. Цей клас включає в себе такі функції:

- load\_dataset(); проводиться попередня обробка та розподіляються дані на тренувальний та тестовий набори; завантажені дані потім повертаються;
- build\_model(self); конструюється архітектура моделі CNN для передбачення емоцій;
- train\_model(self, X\_train, X\_test, Y\_train, Y\_test); тренує побудовану модель, використовуючи зазначені дані для тренування та тестування;
- save\_model(self, file); зберігає навчену модель у вказаний файл.

#### 4.1.2 Клас EmotionProcessingHelper

Основна задача класу EmotionProcessingHelper- попередня обробка вхідних даних (зображень) та прогнозування емоції за допомогою попередньо збереженої моделі. Цей клас включає в себе такі функції:

- `preprocess_image(self, image_file)`; обробляє зображення, виконує передпроцесинг, і повертає оброблений масив зображення для подальшого прогнозування емоцій;
- `predict_emotion(self, image_array)`; Прогнозує емоцію на основі обробленого масиву зображення за допомогою попередньо навченої моделі розпізнавання емоцій.

#### 4.1.3 Клас DBHelper

Клас DBHelper – забезпечує доступ до запису/читання даних БД. Цей клас включає в себе такі функції:

- `save_prediction_to_db(self, user_id, user_label, image)`; зберігає дані прогнозу емоцій користувача у базі даних. Включає інформацію про ідентифікатор користувача, прогнозовану мітку емоції та зображення, а також дату створення запису;
- `get_user_history_last_7_days(self, user_id)`; повертає історію прогнозів емоцій користувача за останні 7 днів; ключає дані про ідентифікатор користувача, прогнозовану мітку емоції, зображення та дату створення запису.

#### 4.1.4 Клас UserHandler

Клас UserHandler – клас, що обробляє запити користувача, пов'язані із реєстрацією та входом у систему. Він включає в себе такі функції:

- `signup(self, data)`; реєструє нового користувача, перевіряючи унікальність імені та електронної адреси.
- `login(self, data)`; виконує вхід користувача, перевіряючи вірність введеного паролю.

#### 4.1.5 Клас EmotionsHandler

Клас EmotionsHandler – клас, який обробляє запити, пов'язані з передбаченням емоцій на основі зображень та взаємодіє з базою даних.

Цей клас включає в себе такі функції:

- emotion\_prediction\_response (self current\_user\_id user\_label, predicted\_emotion); створює відповідь на запит передбачення емоції;
- predict\_emotion(self, data, current\_user\_id); передбачає емоцію на основі отриманого зображення, зберігає результат у базі даних та повертає відповідь із передбаченою емоцією та ідентифікатором користувача.

#### 4.1.6 Клас HistoryHandler

Клас HistoryHandler – обробляє запити, пов'язані із отриманням історії прогнозів емоцій користувача.

Цей клас включає в себе такі функції:

- get\_user\_history(self, current\_user\_id); отримує історію прогнозів емоцій користувача за останні 7 днів та повертає відповідь з отриманою історією.

#### 4.1.7 Клас App

Клас App – це головний клас програми, де описані всі ендпоїнти. Цей клас включає в себе такі функції:

- \_\_init\_\_(self, port=8000, debug=True); ініціалізує об'єкт App з параметрами порту та відлагодження; задає конфігураційні параметри Flask, ініціалізує необхідні класи (наприклад, EmotionProcessingHelper, UserHandler, DBHelper, EmotionsHandler, HistoryHandler);
- setup\_routes(self); встановлює маршрути для різних ендпоїнтів, таких як реєстрація, вхід, передбачення емоцій та отримання історії;

– `run(self, port, debug)`; запускає вебзастосунок із вказаними параметрами порту та відлагодження.

#### 4.1.8 Ендпоїнти застосунку

У класі `App` була зазначена функція, що ініціалізує ендпоїнти. Їх детальний опис представлено в табл. 4.1.

Таблиця 4.1 – Основні ендпоїнти застосунку

| Маршрут                       | Метод | Функція обробки                                | Опис                                                                                                 |
|-------------------------------|-------|------------------------------------------------|------------------------------------------------------------------------------------------------------|
| <code>/signup</code>          | POST  | <code>UserHandler.signup()</code>              | Обробляє запит на реєстрацію нового користувача, перевіряє унікальність імені та електронної адреси. |
| <code>/login</code>           | POST  | <code>UserHandler.login()</code>               | Обробляє запит на вхід користувача до системи, перевіряє вірність введеного паролю.                  |
| <code>/predict_emotion</code> | POST  | <code>EmotionsHandler.predict_emotion()</code> | Обробляє запит на передбачення емоцій на основі надісланого зображення.                              |
| <code>/user_history</code>    | GET   | <code>HistoryHandler.get_user_history()</code> | Обробляє запит на отримання історії прогнозів емоцій користувача за останні 7 днів.                  |

### 4.2 Основні рішення щодо розробки мобільного застосунку

#### 4.2.1 Файлова структура застосунку

Обрана архітектура застосунку дає можливість виокремити бізнес-логіку (*use-cases*) від шару представлення. За зв'язок з ендпоїнтами та внутрішнім середовищем відповідає директорія `data`. Всі моделі зберігаються в `models`. Елементи інтерфейсу і екрани розташовані в `presentation`. Бізнес-логіка знаходиться в директорії `use-cases` і розділена на модулі (рис. 4.1).

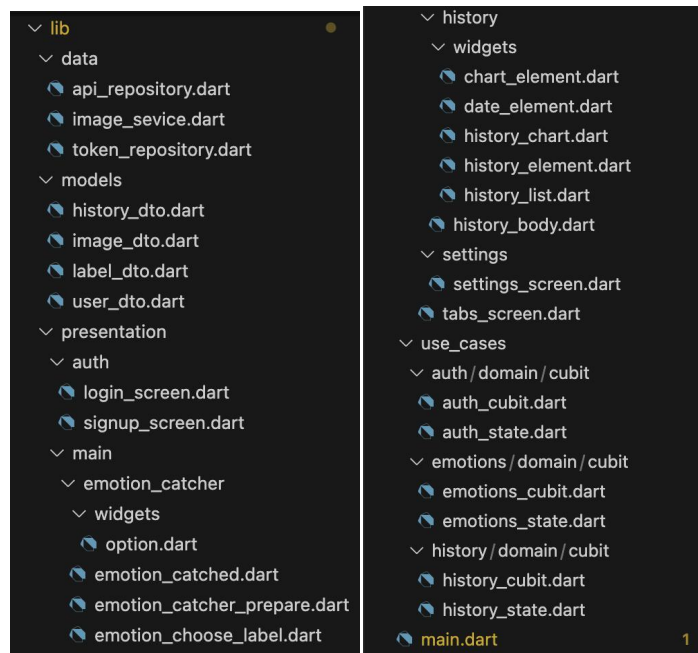


Рисунок 4.1 - Файлова структура мобільного застосунку

#### 4.2.2 Data Layer

ApiRepository є основним класом-репозиторієм для доступу до бекенд-застосунку. Він використовує бібліотеку http для відправлення запитів. В ньому реалізовані такі функції:

- signup(UserDTO userDTO); відправляє запит на реєстрацію користувача до API;
- login(UserDTO userDTO); відправляє запит на авторизацію до API;
- predictEmotion(ImageDTO imageDTO); відправляє запит на передбачення емоції на основі зображення;
- getUserHistory(); отримує історію передбачень емоцій користувача.

Також є клас TokenRepository, через який відбувається збереження токена на девайсі користувача. Доступ реалізовано за допомогою бібліотеки hive і створеного інтерфейсу HiveStorage. Функції класу:

- take(); взяти значення, записане в хранилищі;
- save(); зберегти нове значення токена;
- delete(); видалити токен при виході з акаунту.

Для отримання зображень і використання нативних функцій був створений клас `ImageService`. Функції цього класу:

- `getCameraImage()`; функція для отримання зображення за допомогою камери;
- `getGalleryImage()`; функція для отримання зображення із сховища телефону.

Так як вся обробка зображення відбувається на сервері – в мобільному застосунку відправляється оригінальний файл у API request.

### 4.2.3 Domain Layer

В шарі Domain представлена бізнес-логіка, яка розроблена за допомогою архітектури state management - Bloc. Для всіх use-cases були створені окремі директорії і для кожного модулю були створені такі сутності - Cubit і State. В парадигмі Bloc є два підходи: Bloc і Cubit. Було обрано останній, так як Cubit має більш простий стиль написання і підходить для невеликих обсягів логіки застосунку. За допомогою асинхронних функцій Cubit може оголошувати нові стани – States. Під час розробки застосунку було створено три Cubit для основних модулів. AuthCubit відповідає за блок авторизації і має наступні функції:

- `init()`; викликається в момент запуску застосунку, отримується значення токена з локального сховища і авторизує або ні користувача;
- `login(email, password)`; авторизація користувача за email і паролем;
- `signup(username, email, password)`; реєстрація користувача за іменем користувача, email і паролем;
- `logout()`; вихід з акаунту.

EmotionsCubit відповідає за блок розпізнавання емоцій і має наступні функції:

- `predictEmotion(image, String userLabel)`; відправка запиту на передбачення емоції з заданим попередньо значенням юзера.

HistoryCubit відповідає за блок історії розпізнавань емоцій і має наступні функції:

- loadData(); відправка запиту списку всієї історії користувача.

### 4.3 Структура бази даних

MongoDB є документо-орієнтованою системою керування базами даних, отже вона не використовує традиційні таблиці, а замість цього вона використовує колекції, які містять документи у форматі JSON-подібних об'єктів. У застосунку існують дві колекції: "users" і "predictions". Перша містить в собі дані про зареєстрованих юзерів. Детальний опис полів наведено в табл. 4.2. ObjectId - унікальний ідентифікатор MongoDB, який автоматично генерується при вставці нового запису.

Таблиця 4.2 – Модель колекції “users”

| Поле     | Тип      | Опис                                 |
|----------|----------|--------------------------------------|
| user_id  | ObjectId | Унікальний ідентифікатор користувача |
| username | String   | Ім'я користувача                     |
| email    | String   | Електронна адреса користувача        |
| password | String   | Хеш паролю користувача               |

Таблиця “predictions” містить в собі дані про розпізнавання користувачів. Детальний опис полів наведено в табл. 4.3.

Таблиця 4.3 – Модель колекції “predictions”

| Поле          | Тип      | Опис                                               |
|---------------|----------|----------------------------------------------------|
| prediction_id | ObjectId | Унікальний ідентифікатор прогнозу                  |
| user_id       | ObjectId | Ідентифікатор користувача, пов'язаний із прогнозом |
| user_label    | String   | Мітка користувача (якщо використовується)          |
| image         | Binary   | Зображення, яке використовується для прогнозу      |
| date_created  | DateTime | Дата та час створення запису прогнозу              |

## **4.4 Алгоритми програми**

### **4.4.1 Алгоритм тренування моделі**

Тренування створеної нейронної мережі відбувається за наступним алгоритмом.

Крок 1. Збір та підготовка даних. Цей крок включає в себе завантаження зображень облич з датасету, що містить категорії емоцій, конвертацію кольорових зображень в чорно-білі та зменшення розміру до (48, 48) для скорочення обчислювальної потужності.

Крок 2. Розділення даних на тренувальний та тестовий набори та кодування назв категорій.

Крок 3. Визначення форми вхідного шару, який відповідає розміру та формату вхідних зображень. Наприклад, (48, 48, 1) для чорно-білих зображень розміру 48x48 пікселів.

Крок 4. Додавання кількох згорткових шарів для виявлення різних аспектів зображень, активація ReLU для введення нелінійності, використання dropout для попередження перенавчання.

Крок 5. Додавання пулінгових шарів (MaxPooling) для зменшення розмірів зображення та виділення важливих ознак.

Крок 6. Додавання повністю з'єднаних шарів, які об'єднують інформацію та готують її для виходу. Використання активації ReLU та dropout.

Крок 7. Визначення вихідного шару з кількістю нейронів, що відповідає кількості класів, а також використання softmax для отримання ймовірностей кожного класу.

Крок 8. Компіляція моделі. Обирається оптимізатор (Adam) для налаштування ваг моделі під час тренування, функція втрат (категоріальна крос-ентропія) для задач класифікації, метрики для вимірювання точності.

Крок 9. Тренування моделі. Здійснюється на тренувальному наборі за допомогою методу fit з використанням 50 епох.

Крок 10. Збереження навченої моделі для подальшого використання.

#### **4.4.2 Алгоритм розпізнавання зображення**

В мобільному застосунку реалізоване розпізнавання зображень. Розглянемо комплексний алгоритм цієї функції.

Крок 1. Авторизація користувача в застосунку.

Крок 2. Вибір джерела зображення: камера чи сховище телефону.

Крок 3. Отримання зображення відкористувача.

Крок 4. Вибір користувачем категорії, до якої на його думку відноситься ця емоція або пропуск цього етапу.

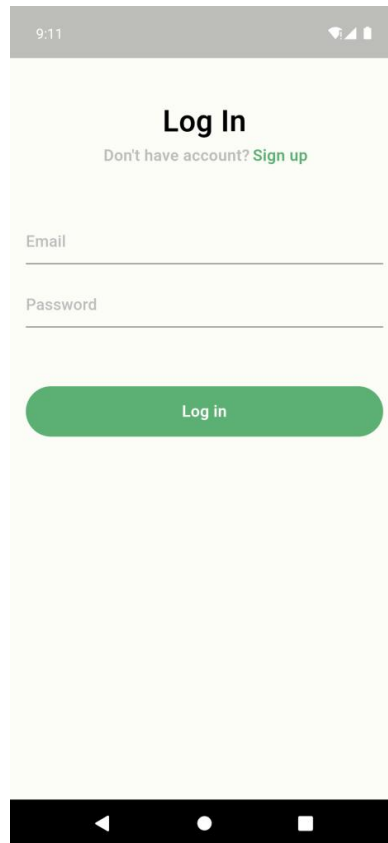
Крок 5. Відправка зображення та категорії запитом на сервер, отримання значення емоції, яку спрогнозувала модель.

Крок 6. Вивід користувачу зображення, прогнозованої емоції та емоції, що обрав користувач для наочного порівняння.

Крок 7. Збереження даних в історії, оновлення сторінки історії емоцій.

#### **4.4 Опис розроблених інтерфейсів**

Інтерфейс розробленого застосунку має бути лаконічним, не перевантаженим елементами, простий та зрозумілий для використання. Користування чітким та однозначним через загально відомі і використовувані компоненти (наприклад, кнопки, текстові поля, іконки т.д.). Інтерфейси були розроблені з використанням 2 основних кольорів, шрифт був обраний Inter. При відкритті програми неавторизованого користувача доступними є екран авторизації (рис. 4.2) та екран реєстрації (рис. 4.3).



9:11

## Log In

Don't have account? [Sign up](#)

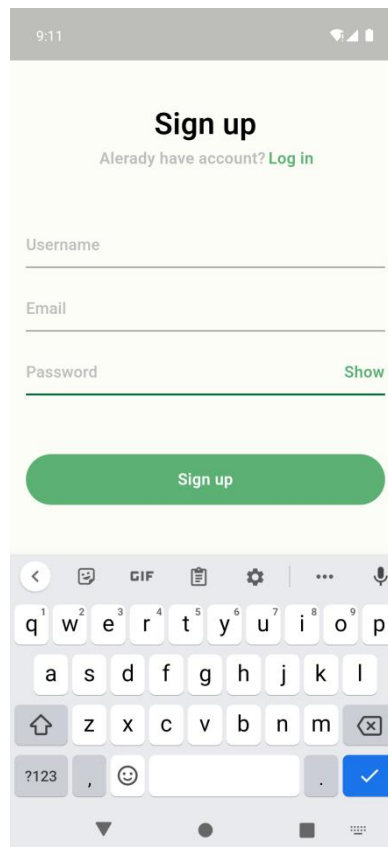
Email

Password

Log in

This is a mobile app login screen. It features a light yellow background. At the top, there's a status bar with the time 9:11 and signal icons. Below the title 'Log In', there's a link 'Sign up' for users who don't have an account. There are two input fields for 'Email' and 'Password'. A green 'Log in' button is positioned below the password field. The bottom of the screen shows a black navigation bar with standard Android icons.

Рисунок 4.2 – Сторінка авторизації



9:11

## Sign up

Alerady have account? [Log in](#)

Username

Email

Password [Show](#)

Sign up

This is a mobile app sign-up screen. It has a light yellow background. The status bar at the top shows 9:11 and signal icons. The title is 'Sign up', followed by a link 'Log in' for users who already have an account. There are three input fields: 'Username', 'Email', and 'Password'. The 'Password' field has a 'Show' link to toggle visibility. A green 'Sign up' button is located below the password field. A virtual keyboard is visible at the bottom of the screen, indicating the password field is active.

Рисунок 4.3 - Сторінка реєстрації

Після авторизації користувачу доступний основний функціонал. Головна навігація застосунку відбувається через переключення табів в елементі BottomAppBar. Для розпізнавання емоції на першому табі треба обрати джерело зображення (рис. 4.4). При натисканні на кнопку відкривається нативна програма камери або галереї.

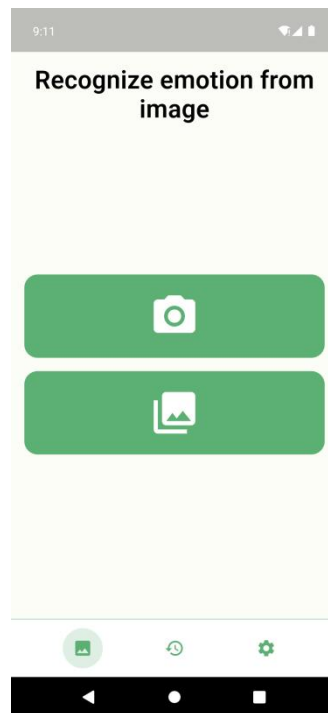


Рисунок 4.4 – Головна сторінка з вибором джерела зображень

Після вибору зображення відкривається сторінка вибору емоції (рис 4.5) зі списком елементів RadioButton і двома кнопка - для вибору або пропуску цього кроку.

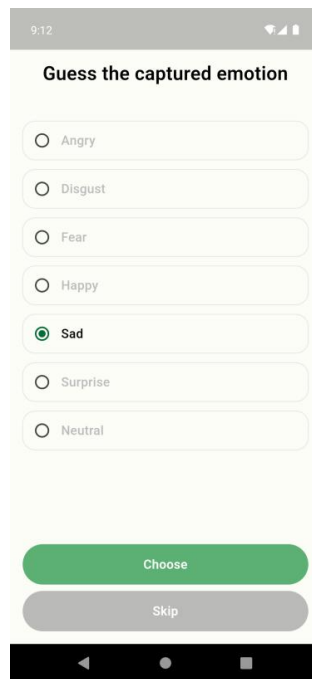


Рисунок 4.5 – Сторінка вибору емоції

Далі виводиться отримана з бекенду емоція і зіставлення її з попередньо обраною (рис. 4.6). По натисканню на кнопку finish користувач повертається на головний екран.

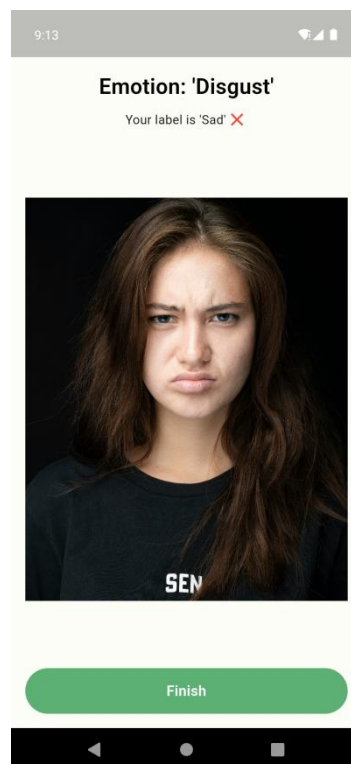


Рисунок 4.6 – Результат розпізнавання емоції на зображенні

Також користувач може перейти через таби навігації на сторінку історії. В верхній частині розташований графік, який показує успішність користувача за останній тиждень у відсотковому співвідношенні на прикладі діаграми. Нижче розташована вся історія, яка включає зображення та категорії, до яких їх було віднесено (рис.4.7).

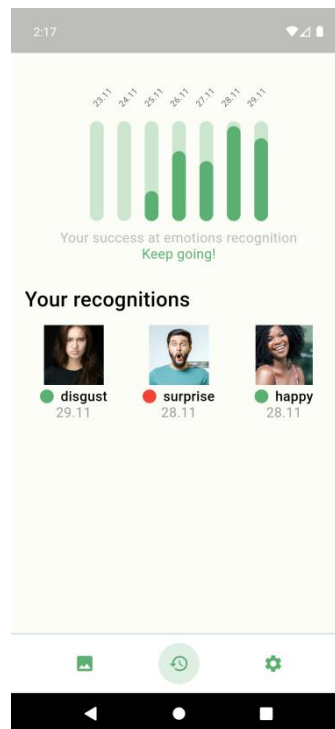


Рисунок 4.7 – Сторінка історії емоцій

Завершальним є екран налаштувань, де можна переглянути, який користувач зараз авторизований та зробити вихід з акаунту (рис. 4.8).

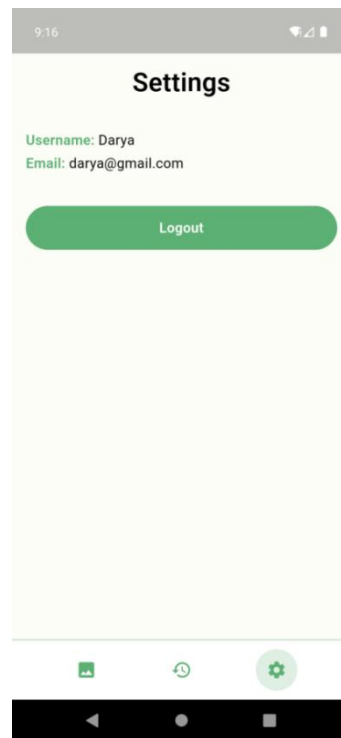


Рисунок 4.8 – Сторінка налаштувань

#### 4.5 Висновки за розділом 4

В даному розділі були розглянуті основні рішення щодо реалізації бекенд і клієнт частин. Були проаналізовані створені класи, а також виділені основні алгоритми в роботі програми. Беручи за основу алгоритми, були створені основні інтерфейси застосунку. Також була описана обрана база даних та були описані її основні колекції.

## **5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПРЕМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ**

### **5.1 Призначення та умови застосування програми**

Призначенням мобільного застосунку є розпізнавання емоцій для людей з розладами аутистичного спектру або для людей які вдосконалюють навички розпізнавання емоцій. Користувачі можуть створювати свої акаунти, робити фотографії, отримувати результати розпізнавання, переглядати історію своїх дій та оцінювати прогрес.

Для правильної роботи програми важливо враховувати наступні програмно-технічні умови:

- девайс з ОС Android 9.0 і вище або iOS 11.0 і вище;
- оперативна пам'ять мінімум 1 ГБ;
- вільний простір не менше 100 МБ;
- інтернет-підключення;
- розгорнутий локально чи віддалено сервер.

Для розгортання серверу необхідні наступні умови:

- ОС Windows 10 або новіша, або MacOS версії 13 і вище;
- оперативна пам'ять мінімум 4 ГБ;
- вільний простір на комп'ютері не менше 8 ГБ;
- процесор мінімум 1 ГГц частота;
- встановлене програмне забезпечення (наприклад VSCode);
- інтернет-підключення;
- версія Python найменше 3.8.0.

### **5.2 Виконання програми**

Для роботи застосунку необхідно запустити бекенд на локальному чи віддаленому сервері та зробити білд мобільного застосунку.

Для успішного запуску проекту необхідно створити віртуальне середовище розробки і активувати його за допомогою команд:

```
python -m venv venv  
source venv/bin/activate
```

Також необхідно встановити всі бібліотеки, що використовуються в цьому середовищі за допомогою команди:

```
pip install [library]
```

Для запуску бекенду треба перейти в директорію, де знаходиться файл `app.py`, відкрити термінал або консоль і ввести наступну команду:

```
python app.py
```

Бекенд Flask буде запущений локально за адресою `http://127.0.0.1:8000/`, і буде можливість взаємодіяти з ним через локальний веббраузер, API або інші засоби зв'язку.

Для запуску мобільного застосунку першочергово треба завантажити всі потрібні бібліотеки командою

```
flutter pub get
```

Далі в файлі `api_repository.dart` необхідно взяти адресу (локальну чи віддалену) сервера для використання Representational State Transfer API (REST API) в роботі застосунку. Для запуску застосунку на Android або iOS девайсі чи емуляторі треба виконати наступну команду:

```
flutter run --release
```

Першою сторінкою буде сторінка авторизації. На цьому екрані є валідація полів на непусті значення, а на екрані реєстрації - валідація email на правильність вводу за маскою. Якщо значення введено неправильно - виводиться відповідне повідомлення (рис. 5.1). Якщо реєстрація відбувається за email, який вже зареєстровано – з'являється наступне повідомлення (рис. 5.2).

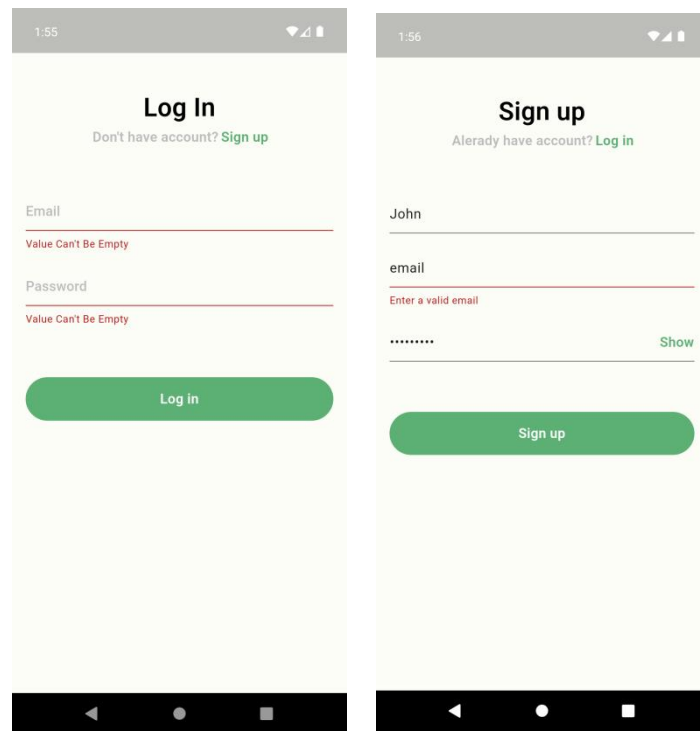


Рисунок 5.1 – Валідація значень в текстових полях

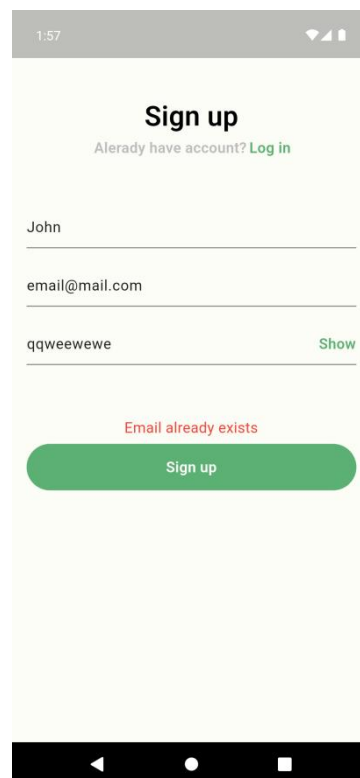


Рисунок 5.2 – Помилка серверу, якщо користувач уже є в системі

Після введення коректних даних на сторінці авторизації відкривається головна сторінка. Для вибору зображення необхідно дати дозвіл на використання камери і/або галереї девайсу (рис.5.3).

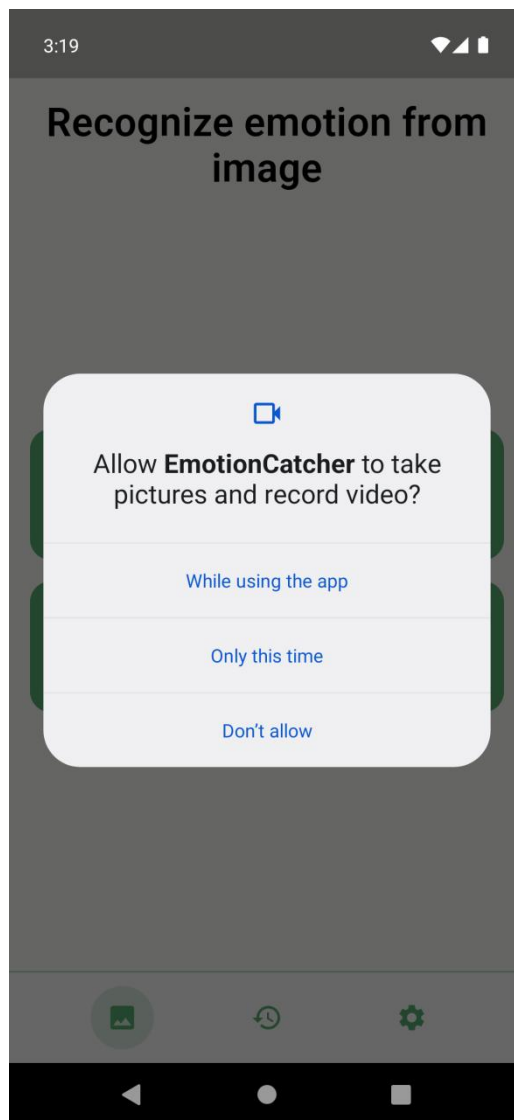


Рисунок 5.3 – Приклад діалогу для надання дозволу використовувати камеру

Щоб отримати результат розпізнавання зображення, необхідно мати доступ до інтернету; якщо на якомусь етапі пропадає інтернет - буде виведено повідомлення користувачу (рис. 5.4).

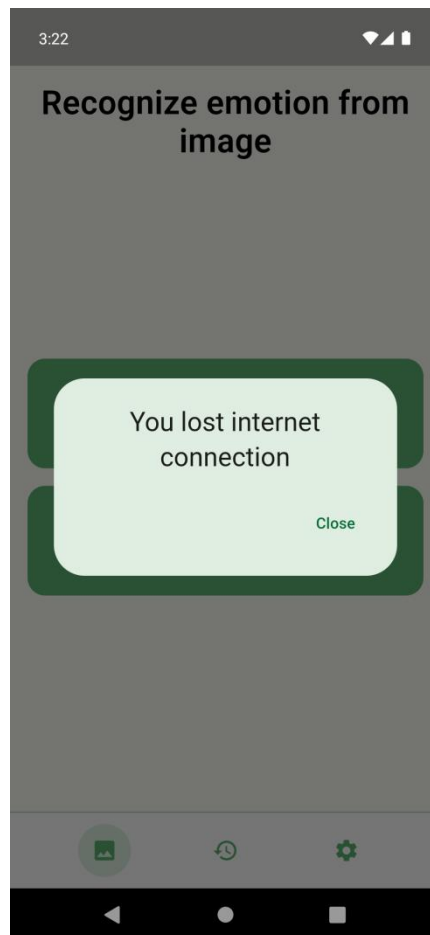


Рисунок 5.4 – Повідомлення про втрату інтернет-з'єднання

Також є обробка порожніх значень, як наприклад пустого списку історії. В такому випадку юзер буде бачити, що історія ще пуста (рис. 5.5).



Рисунок 5.5 – Порожня історія користувача

### 5.3 Тестування програми

З огляду на те що програма на початковому етапі не планується використовуватись у великих масштабах, тестування застосунку відбуватиметься в більшій мірі вручну, але також з використанням автоматизованих тестів.

У застосунках Flutter є 3 типи тестування:

- unit test, перевіряє одну функцію, метод або клас;
- widget test, перевіряє один віджет;
- integration test, перевіряє програму в цілому або велику частину програми [25].

Сервіси і на cubits вирішено тестувати за допомогою unit test, частини екранів і віджет за допомогою widget tests. Більш обширні частини програми будуть тестуватися вручну за допомогою створених тест-кейсів.

### 5.3.1 Unit tests

Приклад unit test буде показано на основі AuthCubit. Для тестування всіх випадків був складений чекліст, за яким розроблялись тести.

Таблиця 5.1 – Приклад роботи unit тестів

| Тест           | Вихідний стан    | Дії                                           | Очікуваний результат  |
|----------------|------------------|-----------------------------------------------|-----------------------|
| Init           | AuthInitialState | Викликати метод init()                        | Еміт AuthInitialState |
| Login Success  | Authenticated    | Викликати метод login з правильними даними    | Еміт Authenticated    |
| Login Failure  | AuthInitialState | Викликати метод login з неправильними даними  | Еміт AuthInitialState |
| Signup Success | Authenticated    | Викликати метод signup з правильними даними   | Еміт Authenticated    |
| Signup Failure | AuthInitialState | Викликати метод signup з неправильними даними | Еміт AuthErrorState   |

### 5.3.2 Widget тести

Для віджет тестування необхідно скласти шаблон, за яким воно буде відбуватись.

1. Створити мок значень (імітація реальних даних).
2. Додати бажаний віджет.
3. Переконавшись, що він відображається без будь-яких помилок.
4. Перевірити початковий стан завантаження.
5. Перевірити відображення завантажених даних, чи вони правильні.

Так було виконано тестування наступний віджетів: PrepareEmotionWidget, HistoryWidget, EmotionUserLabel, EmotionRecognizedWidget, LoginBody, SignupBody.

### 5.3.3 Результати тестування

Щоб запустити тестування достатньо виконати програму в консолі:

```
flutter test test/main_test.dart
```

Результат проходження тестів зображено на рис.5.6.

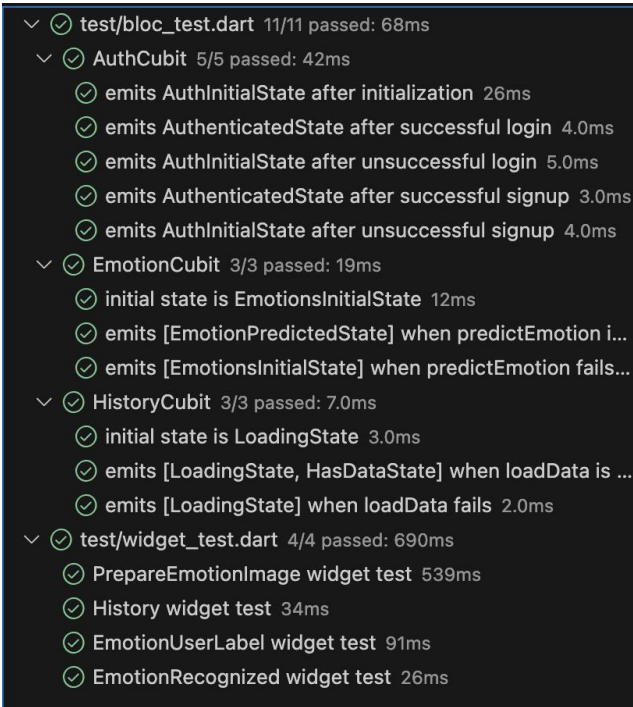


Рисунок 5.6 – Результат проходження юніт і віджет тестування

### 5.3.4 Ручне тестування за допомогою тест-кейсів

Тестовий випадок, тест-кейс— це артефакт, що описує сукупність кроків, конкретних умов та параметрів, необхідних для перевірки реалізації функції, що тестується чи її частини. Під тест кейсом йдеться про наступну структуру: Action > Expected Result[26].

Тест кейси були описані для функціоналу розпізнавання емоції (табл. 5.2) і перегляду історії (табл. 5.3).

Таблиця 5.2 – Тест-кейси функціоналу розпізнавання емоції

| № | Опис тест-кейсу            | Передумови                             | Кроки виконання                         | Очікуваний результат                                       |
|---|----------------------------|----------------------------------------|-----------------------------------------|------------------------------------------------------------|
| 1 | 2                          | 3                                      | 4                                       | 5                                                          |
| 1 | Вибір зображення з камери  | Користувач на головному екрані додатку | 1. Торкніться кнопки з іконкою Камера.  | Відкриється нативна програма камери для зйомки зображення. |
| 2 | Вибір зображення з галереї | Користувач на головному екрані додатку | 1. Торкніться кнопки з іконкою Галерея. | Відкриється галерея для вибору зображення.                 |

Продовження таблиці 5.2

| 1 | 2                     | 3                               | 4                                                                                                                                    | 5                                                                                                         |
|---|-----------------------|---------------------------------|--------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| 3 | Вибір емоції          | Користувач вже обрав зображення | 1. Після вибору зображення натисніть на кнопку підтвердження. 2. Оберіть одну з емоцій зі списку RadioButton. 3. Натисніть "Choose". | Відобразиться вибрана емоція, емоція яку спрогнозувала модель, зображення користувача та кнопка "Finish". |
| 4 | Пропуск вибору емоції | Користувач вже обрав зображення | 1. Після вибору зображення натисніть на кнопку підтвердження. 2. Натисніть "Skip"                                                    | Відобразиться кнопка "Finish" без обраної емоції.                                                         |

Таблиця 5.3 – Тест-кейси функціоналу перегляду історії

| № | Опис тест-кейсу             | Передумови                             | Кроки виконання                                                      | Очікуваний результат                                          |
|---|-----------------------------|----------------------------------------|----------------------------------------------------------------------|---------------------------------------------------------------|
| 1 | Перехід на сторінку історії | Користувач на головному екрані додатку | 1. Торкніться табу з іконкою історії.                                | Перейдете на сторінку історії.                                |
| 2 | Перегляд графіку            | Користувач на сторінці історії         | 1. Оцініть графік в верхній частині сторінки.                        | Відобразиться графік успішності за останній тиждень.          |
| 3 | Перегляд історії зображень  | Користувач на сторінці історії         | 1. Прокрутіть сторінку вниз для перегляду всієї історії.             | Відобразиться вся історія, включаючи зображення та категорії. |
| 4 | Обробка порожньої історії   | Користувач на сторінці історії         | 1. Очистіть історію (якщо можливо). 2. Перегляньте сторінку історії. | Відобразиться повідомлення про порожню історію.               |

## 5.4 Висновки за розділом 5

У цьому розділі була проведена обробка помилок в застосунку і описаний порядок виконання програми. Також було визначено стратегію тестування для застосунку "EmotionCatcher". Вирішено використовувати комбінацію ручного та автоматизованого тестування, обираючи unit test та widget test в межах фреймворку Flutter. Це дозволило перевірити різні рівні застосунку, забезпечуючи його стабільність та надійність.

## ВИСНОВКИ

Під час виконання дипломної роботи було проведено ретельне вивчення предметної області - розпізнавання емоцій у людей з розладами аутистичного спектру. Досліджено ключові аспекти цієї області та розглянуті основні вирішення, зокрема створення програмного забезпечення "EmotionCatcher" для зручного та ефективного розпізнавання емоцій у цільової аудиторії.

У першому розділі були розглянуті існуючі аналоги та методи розпізнавання емоцій, аналіз яких допоміг сформулювати основні вимоги до розроблюваного проекту. Окрім того, визначено переваги та недоліки існуючих рішень, які слугували основою для подальшого вдосконалення "EmotionCatcher".

У другому розділі "Матеріали і методи" дипломної роботи проведено детальний аналіз використаних матеріалів та методів для розробки та оптимізації нейронної мережі в рамках проекту "EmotionCatcher".

Третій розділ присвячений вибору технічних засобів та інструментів розробки. Обрано мову програмування, фреймворк та середовище розробки, що оптимально відповідають поставленим завданням. Також обрано систему керування станами та сформовано компонентну діаграму системи.

В четвертому розділі розглянута логіка роботи програми, розроблені алгоритми та описано модулі системи розпізнавання емоцій. Детально висвітлено структуру бази даних та основні сутності, що використовуються у проекті. І в останньому розділі було описано вимоги до системи та протестовано її.

В результаті було успішно розроблено програмне забезпечення "EmotionCatcher", яке відповідає усім поставленим завданням та функціональним вимогам. Реалізовано можливість ефективного розпізнавання емоцій та надання користувачам зручного інструменту для покращення соціальної взаємодії та якості життя. Можливості для

подальшого розвитку включають розширення функціоналу, покращення інтерфейсу та додаткові можливості адаптації для конкретних потреб користувачів.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Nagy E. Atypical Facial Emotion Recognition in Children with Autism Spectrum Disorders: Exploratory Analysis on the Role of Task Demands [Text] / E. Nagy, L. Prentice, T. Wakeling // Perception. – 2021. – Vol. 50(9). – P. 819–833.
2. Shalev-Shwartz , S. Understanding Machine Learning: From Theory to Algorithms [Text] / S. Shalev-Shwartz, S. Ben-David. – NewYork: CambridgeUniversityPress, 2014. – 449 p.
3. Murphy K. Machine Learning: A Probabilistic Perspective [Text]/ K. Murphy. – London: Massachusetts Institute of Technology, 2012. – 1067 p.
4. Lee, K. What is Image classification? Techniques and Examples [Electronic resource] / K. Lee. – Access mode: <https://kili-technology.com/data-labeling/computer-vision/image-annotation/what-is-image-classification#semi-supervised-learning>
5. Mehendale, N. Facial emotion recognition using convolutional neural networks (FERC) / N. Mehendale // SN Applied Sciences –2020. –Vol. 2. –P. 446.
6. Sanghvi, K. Image Classification Techniques [Electronic resource] / K. Sanghvi. – Access mode: <https://medium.com/analytics-vidhya/image-classification-techniques-83fd87011cac>
7. Image Classification [Electronic resource]. – Access mode: <https://ltb.itc.utwente.nl/509/concept/89060>
8. When to Use MLP, CNN, and RNN Neural Networks [Electronic resource]. – Access mode: <https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/>
9. Brownlee, J. Multilayer Perceptron (MLP) vs Convolutional Neural Network in Deep Learning [Electronic resource] / J. Brownlee. – Access mode: <https://medium.com/data-science-bootcamp/multilayer-perceptron-mlp-vs-convolutional-neural-network-in-deep-learning-c890f487a8f1>
10. What are convolutional neural networks? [Electronic resource]. – Access mode: <https://www.ibm.com/topics/convolutional-neural-networks>

11. FaceAnalyzer :Age, Gender [Electronic resource]. – Access mode: <https://apps.apple.com/ua/app/faceanalyzer-age-gender/id1267719377?l=uk?l=uk>
12. Emotimeter - Emotion detector [Electronic resource]. – Access mode: [https://play.google.com/store/apps/details?id=com.reaimagine.josem.emotimeter\\_facialemotionrecognizer&hl=uk&gl=US](https://play.google.com/store/apps/details?id=com.reaimagine.josem.emotimeter_facialemotionrecognizer&hl=uk&gl=US)
13. Koech, K.E. Cross-Entropy Loss Function [Electronic resource] / K.E. Koech – Access mode: <https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e>
14. Shah, D. Cross Entropy Loss: Intro, Applications, Code [Electronic resource] / D. Shah. – Access mode: <https://www.v7labs.com/blog/cross-entropy-loss-guide>
15. Bruballa, R.G. Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names [Electronic resource] / R.G. Bruballa – Access mode: [https://gombru.github.io/2018/05/23/cross\\_entropy\\_loss/](https://gombru.github.io/2018/05/23/cross_entropy_loss/)
16. Gupta, A. A Comprehensive Guide on Optimizers in Deep Learning [Electronic resource] / A. Gupta – Access mode: [https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/#Adam\\_Deep\\_Learning\\_Optimizer](https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/#Adam_Deep_Learning_Optimizer)
17. Java (programming language) [Electronic resource]. – Access mode: [https://en.wikipedia.org/wiki/Java\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Java_(programming_language))
18. Vasist, K. Deep Learning in Java [Electronic resource] / K. Vasist. – Access mode: <https://towardsdatascience.com/deep-learning-in-java-d9b54ae1423a>
19. JavaScript [Electronic resource]. – Access mode: <https://en.wikipedia.org/wiki/JavaScript>
20. Thomas, J. Machine Learning In Node.js With TensorFlow.js [Electronic resource] / J. Thomas . – Access mode: <https://jamesthom.as/2018/08/machine-learning-in-node.js-with-tensorflow.js/>
21. Python (programming language) [Electronic resource]. – Access mode: [https://en.wikipedia.org/wiki/Python\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))

22. Pruciak, M. Python vs JavaScript: Which Is Better For Machine Learning Project? [Electronic resource] / M. Pruciak. – Access mode: <https://www.ideamotive.co/blog/python-vs-javascript>

23. Montaña, D. Why Use Flutter: Pros And Cons Of Flutter App Development [Electronic resource] / D. Montaña. – Access mode: <https://waverleysoftware.com/blog/why-use-flutter-pros-and-cons/>

24. Nayak, S.K. Flutter Development: VS Code vs. Android Studio [Electronic resource] / S.K. Nayak. – Access mode: <https://shivamkumarnayak.medium.com/flutter-development-vs-code-vs-android-studio-9e93f4917cc>

25. Testing Flutter apps [Electronic resource]. – Access mode: <https://docs.flutter.dev/testing/overview>

26. Тестовий випадок [Electronic resource]. – Access mode: [https://uk.wikipedia.org/wiki/Тестовий\\_випадок](https://uk.wikipedia.org/wiki/Тестовий_випадок)

**ДОДАТОК А**  
**Технічне завдання**

## **A.1 Вступ**

### **A.1.1 Назва програми**

Мобільний застосунок “EmotionCatcher”.

### **A.1.2 Призначення та область застосування**

Програмне забезпечення розробляється для людей з розладами аутистичного спектру. Розроблене програмне забезпечення стане в нагоді дітям і дорослим, які мають проблеми з розпізнавання емоцій.

## **A.2 Підстави для розробки**

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація мобільного застосунку для розпізнавання емоцій за зображенням» затверджене наказом номер 437 від 14 листопада 2023 р. по Національному університету "Запорізька політехніка".

## **A.3 Призначення розробки**

Програмне забезпечення призначене для розпізнавання емоцій на зображеннях, створення статистики успішності розпізнавань користувача і системи.

## **A.4 Вимоги до програми**

### **A.4.1 Вимоги до функціональних характеристик**

Програмне забезпечення “EmotionCatcher” має виконувати наступні функціональні задачі:

- реєстрація користувача;
- авторизація користувача;
- розпізнавання емоції за зображенням;
- перегляд історії розпізнавань;
- вихід з акаунту.

#### **A.4.2 Вимоги до надійності**

Система повинна розпізнавати ситуації нерозуміння системою користувача (користувачем програми), а також мати обробку збійних ситуацій. Наприклад, якщо користувач вводить дані входу, які вже є в БД, повинно виводитись повідомлення, що такий користувач існує. Також, якщо зображення не було відправлене через поганий інтернет-зв'язок, має з'явитись відповідна реакція застосунку. Перед виходом з акаунту, повинно виводитись повідомлення про підтвердження.

#### **A.4.3 Вимоги до технічних та програмних засобів**

Для забезпечення коректного функціонування ПЗ у користувача має бути смартфон з такими характеристиками:

- ОС Android 9.0 і вище;
- оперативну пам'ять об'ємом, 1Гігабайт, не менше;
- вільну пам'ять 100 Мб, не менше.

#### **A.5 Стадії і етапи розробки**

Розробка повинна бути проведена в чотири стадії.

Розробка технічного завдання.

Стадія «Розробка технічного завдання» складається з наступних етапів:

- розробка технічного завдання;
- узгодження технічного завдання;
- затвердження технічного завдання.

Робоче проектування.

Стадія «Робоче проектування» складається з наступних етапів:

- розробка програми;
- розробка програмної документації;
- випробування програми.

Тестування програми.

Здача роботи в експлуатацію.

На цій стадії повинен бути виконаний етап підготовки і передачі програми.

#### **А.6 Порядок контролю та прийняття**

Дипломна робота повинна відповідати вимогам, які зазначені в пункті А.4 та повинна бути перевірена керівником роботи.

Порядок прийняття роботи відповідає вимогам прийняття магістерських дипломних робіт.

**ДОДАТОК Б**  
**Текст програми**

## EmotionRecognitionModel.py

```

import numpy as np
import tensorflow as tf
import os
import cv2

from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Dropout, Flatten, Dense, MaxPooling2D, Conv2D, Activation, Input
from tensorflow.keras.models import Model
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint
from keras.utils import to_categorical
from keras.models import save_model

class EmotionRecognitionModel:
    def __init__(self, dataset_folder='./CK+', num_of_classes=8):
        self.dataset_folder = dataset_folder
        self.num_of_classes = num_of_classes
        self.model = self.build_model()

    def load_dataset(self):
        subdirectories = os.listdir(self.dataset_folder)
        subdirectories = [item for item in subdirectories if item[0] != "."]

        i = 0
        last = []
        images = []
        labels = []
        temp = subdirectories

        for subdirectory in subdirectories:
            subdirectory_index = temp.index(subdirectory)
            label = subdirectory_index
            path = os.path.join(self.dataset_folder, subdirectory)
            subdirectory_images = os.listdir(path)

            for image_name in subdirectory_images:
                image_path = os.path.join(path, image_name)
                print(image_path + "\t" + str(label))
                image = cv2.imread(image_path)
                image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

```

```

image = cv2.resize(image, (48, 48))
images.append(image)
labels.append(label)
i += 1
last.append(i)

images_x = np.array(images)
labels_y = np.array(labels)
images_x = images_x / 255.0
labels_y_encoded = to_categorical(labels_y, num_classes=self.num_of_classes)

return train_test_split(images_x, labels_y_encoded, test_size=0.25,
random_state=10)

def build_model(self):
input_layer = Input(shape=(48, 48, 1))

conv1 = Conv2D(32, (3, 3), padding='same', strides=(1, 1),
kernel_regularizer=l2(0.001))(input_layer)
conv1 = Dropout(0.1)(conv1)
conv1 = Activation('relu')(conv1)
pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

conv2 = Conv2D(64, (3, 3), padding='same', strides=(1, 1),
kernel_regularizer=l2(0.001))(pool1)
conv2 = Dropout(0.1)(conv2)
conv2 = Activation('relu')(conv2)
pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)

conv3 = Conv2D(128, (3, 3), padding='same', strides=(1, 1),
kernel_regularizer=l2(0.001))(pool2)
conv3 = Dropout(0.1)(conv3)
conv3 = Activation('relu')(conv3)
pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)

conv4 = Conv2D(256, (3, 3), padding='same', strides=(1, 1),
kernel_regularizer=l2(0.001))(pool3)
conv4 = Dropout(0.1)(conv4)
conv4 = Activation('relu')(conv4)
pool4 = MaxPooling2D(pool_size=(2, 2))(conv4)

flatten = Flatten()(pool4)

```

```

dense_1 = Dense(128, activation='relu')(flatten)

drop_1 = Dropout(0.2)(dense_1)

output = Dense(self.num_of_classes, activation="softmax")(drop_1)

return Model(inputs=input_layer, outputs=output)

def train_model(self, model):
    model.compile(optimizer="adam",                loss="categorical_crossentropy",
    metrics=['accuracy'])
    model.summary()

def save_model(self, file='emotion-model.h5'):
    save_model(self.model, file)

emotion_recognition_model = EmotionRecognitionModel()
X_train, X_test, Y_train, Y_test = emotion_recognition_model.load_dataset()
model = emotion_recognition_model.build_model()
emotion_recognition_model.train_model(model)
emotion_recognition_model.save_model()

```

## app.py

```

from datetime import datetime, timedelta

import numpy as np
import tensorflow as tf
from flask import Flask, jsonify, request
from flask_bcrypt import Bcrypt
from flask_jwt_extended import (JWTManager, create_access_token,
get_jwt_identity, jwt_required)
from flask_pymongo import PyMongo
from PIL import Image

from handlers import EmotionsHandler, HistoryHandler, UserHandler
from helpers import EmotionProcessingHelper, DBHelper

class App:
    def __init__(self, port=8000, debug=True):
        self.app = Flask(__name__)
        self.emotion_processing_helper = EmotionProcessingHelper(self.app)
        self.user_handler = UserHandler(

```

```

self.emotion_processing_helper.mongo,
self.emotion_processing_helper.bcrypt,
self.emotion_processing_helper
)
self.db_helper=DBHelper(self.app)
self.emotions_handler = EmotionsHandler(self.emotion_processing_helper)
self.history_handler=HistoryHandler(self.db_helper)

self.app.config['JWT_SECRET_KEY'] = 'dfg45HGt452fs'
self.app.config['MONGO_URI']
'mongodb://admin:admin@localhost:27017/emotionDB'

self.setup_routes()
self.run(port, debug)

def setup_routes(self):
    @self.app.route('/~signup', methods=['POST'])
    def signup():
        return self.user_handler.signup(request.json)

    @self.app.route('/login', methods=['POST'])
    def login():
        return self.user_handler.login(request.json)

    @self.app.route('/predict_emotion', methods=['POST'])
    @jwt_required()
    def predict_emotion():
        current_user_id = get_jwt_identity()

        if 'image' not in request.files:
            return jsonify({"error": "No image file provided"}), 400

        image_file = request.files['image']
        image_array = self.emotion_processing_helper.preprocess_image(image_file)
        predicted_emotion
        self.emotion_processing_helper.predict_emotion(image_array)

        return jsonify({"emotion": predicted_emotion, "user_id": current_user_id})
    @self.app.route('/user_history', methods=['GET'])
    @jwt_required()
    def get_user_history():
        current_user_id = get_jwt_identity()

```

```

user_history
self.history_handler.get_user_history_last_7_days(current_user_id)
return jsonify({"user_history": user_history})

def run(self, port, debug):
self.app.run(port=port, debug=debug)

if __name__ == '__main__':
app_instance = App()

```

## handlers.py

```

from datetime import datetime, timedelta

import numpy as np
import tensorflow as tf
from flask import Flask, jsonify, request
from flask_bcrypt import Bcrypt
from flask_jwt_extended import (JWTManager, create_access_token,
get_jwt_identity, jwt_required)
from flask_pymongo import PyMongo
from PIL import Image

from models import EmotionPredictionDTO, ImageDTO, UserDTO

class UserHandler:
def __init__(self, mongo, bcrypt, emotion_processing_helper):
self.mongo = mongo
self.bcrypt = bcrypt
self.emotion_processing_helper = emotion_processing_helper

def signup(self, data):
user_dto = UserDTO.from_request(data)

if self.mongo.db.users.find_one({"username": user_dto.username}):
return jsonify({"error": "Username already exists"}), 400

if self.mongo.db.users.find_one({"email": user_dto.email}):
return jsonify({"error": "Email already exists"}), 400

hashed_password
self.bcrypt.generate_password_hash(user_dto.password).decode('utf-8')

```

```

user_data = {
    "username": user_dto.username,
    "email": user_dto.email,
    "password": hashed_password,
}
result = self.mongo.db.users.insert_one(user_data)
user_id = str(result.inserted_id)

access_token = create_access_token(identity=user_id)
return jsonify(access_token=access_token), 200

def login(self, data):
    user = self.mongo.db.users.find_one({"email": data['email']})
    user_id = str(user['_id'])

    if user and self.bcrypt.check_password_hash(user['password'],
        data['password']):
        access_token = create_access_token(identity=user_id)
        return jsonify(access_token=access_token), 200
    else:
        return jsonify({"error": "Invalid email or password"}), 401

class EmotionsHandler:
    def __init__(self, emotion_processing_helper, db_helper):
        self.emotion_processing_helper = emotion_processing_helper

    def emotion_prediction_response(self, current_user_id, user_label,
        predicted_emotion):
        emotion_dto = EmotionPredictionDTO(user_id=current_user_id,
            emotion=predicted_emotion)
        return jsonify({"emotion": emotion_dto.emotion, "user_id":
            emotion_dto.user_id})

    def predict_emotion(self, data, current_user_id):
        image_dto = ImageDTO(image_file=request.files['image'])
        user_label = data.get('user_label')

        image_array =
        self.emotion_processing_helper.preprocess_image(image_dto.image_file)
        predicted_emotion =
        self.emotion_processing_helper.predict_emotion(image_array, current_user_id,
            user_label)

```

```

self.db_helper.save_prediction_to_db(current_user_id, user_label, image_array)

return self.emotion_prediction_response(current_user_id, user_label,
predicted_emotion)

class HistoryHandler:
def __init__(self, db_helper):
self.db_helper = db_helper

def get_user_history(self, current_user_id):
history = self.db_helper.get_user_history_last_7_days(current_user_id)
return jsonify({"history": history})

```

## helpers.py

```

from datetime import datetime, timedelta

import numpy as np
import tensorflow as tf
from flask import Flask, jsonify, request
from flask_bcrypt import Bcrypt
from flask_jwt_extended import (JWTManager, create_access_token,
get_jwt_identity, jwt_required)
from flask_pymongo import PyMongo
from PIL import Image

from handlers import EmotionsHandler, HistoryHandler, UserHandler
from models import EmotionPredictionDTO, ImageDTO, UserDTO

class EmotionProcessingHelper:
def __init__(self, app):
self.app = app
self.jwt = JWTManager(app)
self.mongo = PyMongo(app)
self.bcrypt = Bcrypt(app)
self.emotion_model = tf.keras.models.load_model('emotion-model.h5')

def preprocess_image(self, image_file):
image = Image.open(image_file)
image = image.resize((48, 48))
image_array = np.array(image.convert('L')) / 255.0

```

```

image_array = np.reshape(image_array, (1, 48, 48, 1))
return image_array

def predict_emotion(self, image_array):
    prediction = self.emotion_model.predict(image_array)
    emotion_label = np.argmax(prediction)
    emotions = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']
    predicted_emotion = emotions[emotion_label]
    return predicted_emotion

class DBHelper:
    def __init__(self, mongo):
        self.mongo = mongo

    def save_prediction_to_db(self, user_id, user_label, image):
        prediction_data = {
            "user_id": user_id,
            "user_label": user_label,
            "image": image,
            "date_created": datetime.utcnow()
        }
        self.mongo.db.predictions.insert_one(prediction_data)

    def get_user_history_last_7_days(self, user_id):
        start_date = datetime.utcnow() - timedelta(days=7)

        history = self.mongo.db.predictions.find({
            "user_id": user_id,
            "date_created": {"$gte": start_date}
        })

        result_history = []
        for entry in history:
            result_entry = {
                "user_id": entry["user_id"],
                "user_label": entry["user_label"],
                "image": entry["image"],
                "date_created": entry["date_created"]
            }
            result_history.append(result_entry)

        return result_history

```

## api\_repository.dart

```
import 'dart:convert';
import 'package:emotion_catcher_mobile/models/image_dto.dart';
import 'package:emotion_catcher_mobile/models/user_dto.dart';
import 'package:http/http.dart' as http;
import 'package:emotion_catcher_mobile/models/history_dto.dart';
import 'package:emotion_catcher_mobile/models/label_dto.dart';

class ApiRepository {
  final baseUrl = 'http://127.0.0.1:8000';

  Future<Map<String, dynamic>> signup(UserDTO userDTO) async {
    final response = await http.post(
      Uri.parse('$baseUrl/signup'),
      headers: {'Content-Type': 'application/json'},
      body: jsonEncode(userDTO.toJson()),
    );

    if (response.statusCode == 200) {
      return jsonDecode(response.body);
    } else {
      throw Exception('Failed to sign up');
    }
  }

  Future<Map<String, dynamic>> login(UserDTO userDTO) async {
    final response = await http.post(
      Uri.parse('$baseUrl/login'),
      headers: {'Content-Type': 'application/json'},
      body: jsonEncode(userDTO.toJson()),
    );

    if (response.statusCode == 200) {
      return jsonDecode(response.body);
    } else {
      throw Exception('Failed to login');
    }
  }

  Future<LabelDTO> predictEmotion(ImageDTO imageDTO, String token) async {
    final response = await http.post(
      Uri.parse('$baseUrl/predict_emotion'),
```

```

headers: {
  'Content-Type': 'multipart/form-data',
  'Authorization': 'Bearer $token',
},
body: imageDTO,
);

if (response.statusCode == 200) {
  return LabelDTO.fromJson(jsonDecode(response.body));
} else {
  throw Exception('Failed to predict emotion');
}
}

Future<List<HistoryDTO>> getUserHistory(String token) async {
  final response = await http.get(
    Uri.parse('$baseUrl/user_history'),
    headers: {'Authorization': 'Bearer $token'},
  );

  if (response.statusCode == 200) {
    final List<dynamic> historyList = jsonDecode(response.body)['history'];
    return historyList.map((json) => HistoryDTO.fromJson(json)).toList();
  } else {
    throw Exception('Failed to get user history');
  }
}
}

```

### auth\_cubit.dart

```

import 'package:bloc/bloc.dart';
import 'package:emotion_catcher_mobile/data/api_repository.dart';
import 'package:emotion_catcher_mobile/data/token_repository.dart';
import 'package:emotion_catcher_mobile/models/user_dto.dart';
import
'package:emotion_catcher_mobile/use_cases/auth/domain/cubit/auth_state.dart';

class AuthCubit extends Cubit<AuthState> {
  AuthCubit() : super(AuthInitialState());

  void init() {
    if (TokenRepository.take() != null && TokenRepository.take()!.isNotEmpty) {

```

```

emit(AuthInitialState());
}
}

void login(String email, String password) async {
  try {
    final repo = ApiRepository();
    final token = await repo.login(UserDTO(email: email, password: password));
    TokenRepository.save(token['access_token']);
    emit(AuthenticatedState());
  } catch (e) {
    emit(AuthInitialState());
  }
}

void signup(String username, String email, String password) async {
  try {
    final repo = ApiRepository();
    final token = await repo.login(UserDTO(email: email, password: password,
    username: username));
    TokenRepository.save(token['access_token']);
    emit(AuthenticatedState());
  } catch (e) {
    emit(AuthInitialState());
  }
}

void logout() {
  TokenRepository.delete();
  emit(AuthInitialState());
}
}

```

### **auth\_state.dart**

```

abstract class AuthState {}

class AuthInitialState extends AuthState {}

class AuthenticatedState extends AuthState {}

```

### **emotions\_cubit.dart**

```

void predictEmotion(File image, String userLabel) async {

```

```

final repo = ApiRepository();
final token = TokenRepository.take();

final prediction =
await repo.predictEmotion(ImageDTO(label: userLabel, image: image), token ??
'');

emit(EmotionPredictedState(prediction.label, prediction.emotion));
}

```

## history\_chart.dart

```

import
'package:emotion_catcher_mobile/presentation/main/history/widgets/chart_element.dart';
import
'package:emotion_catcher_mobile/presentation/main/history/widgets/date_element.dart';
import 'package:emotion_catcher_mobile/presentation/main/tabs_screen.dart';
import 'package:flutter/material.dart';

class HistoryChart extends StatelessWidget {
const HistoryChart({super.key});
final elements;

@override
Widget build(BuildContext context) {
return Column(
children: [
 SizedBox(height: 40),
 Row(
mainAxisAlignment: MainAxisAlignment.center,
children: [
elements.map((e) => DateElement(date: e.date)),
],
),
 SizedBox(height: 8),
 Row(
mainAxisAlignment: MainAxisAlignment.center,
crossAxisAlignment: CrossAxisAlignment.end,
children: [
elements.map((e) => ChartElement(percent: e.percent)),
],
),
 SizedBox(height: 10),
 Text(
'Your success at emotions recognition',
style: TextStyle(
color: Color(0xFFBDBDBD),
fontSize: 16,
fontFamily: 'Inter',
fontWeight: FontWeight.w400,
height: 0,
),
),
 Text(
'Keep going!',
style: TextStyle(

```

```
color: green,  
fontSize: 16,  
fontFamily: 'Inter',  
fontWeight: FontWeight.w400,  
height: 0,  
) ,  
) ,  
SizedBox(height: 30),  
],  
);  
}  
}
```

## emotion\_catched.dart

```
import 'package:emotion_catcher_mobile/presentation/auth/login_screen.dart';
import 'package:emotion_catcher_mobile/presentation/main/tabs_screen.dart';
import 'package:flutter/material.dart';

class EmotionCaughted extends StatelessWidget {
  const EmotionCaughted({super.key});

  final emotion;
  final label;
  final image;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.center,
            children: [
              Text(
                "Emotion: $emotion",
                textAlign: TextAlign.center,
                style: TextStyle(
                  color: Colors.black,
                  fontSize: 24,
                  fontFamily: 'Inter',
                  fontWeight: FontWeight.w600,
                ),
              ),
              SizedBox(height: 8),
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Text("Your label is label"),
                  Icon(
                    Icons.close,
                    color: Colors.red,
                  )
                ],
              ),
              Expanded(child: Image.asset(image)),
              MainButton(
                text: 'Finish',
                onTap: () {
                  Navigator.of(context).push(MaterialPageRoute(builder: (ctx) => TabsScreen()));
                }
              )
            ],
          ),
        ),
      ),
    );
  }
}
```

```

},
),
],
),
),
),
);
}
}

```

## login\_screen.dart

```

import 'package:emotion_catcher_mobile/presentation/auth/signup_screen.dart';
import 'package:emotion_catcher_mobile/presentation/main/tabs_screen.dart';
import 'package:flutter/material.dart';

class LogIn extends StatelessWidget {
  const LogIn({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SafeArea(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
            children: [
              SizedBox(height: 30),
              const Text(
                'Log In',
                textAlign: TextAlign.center,
                style: TextStyle(
                  color: Colors.black,
                  fontSize: 30,
                  fontFamily: 'Inter',
                  fontWeight: FontWeight.w600,
                  height: 0,
                ),
              ),
              SizedBox(height: 8),
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Text(
                    "Don't have account? ",
                    textAlign: TextAlign.center,
                    style: TextStyle(
                      color: Color(0xFFBDBDBD),
                      fontSize: 16,
                      fontFamily: 'Inter',
                      fontWeight: FontWeight.w600,
                      height: 0,
                    ),
                  ),
                  InkWell(
                    onTap: () {
                      Navigator.of(context).push(MaterialPageRoute(builder: (ctx) => Signup()));
                    },
                    child: Text(
                      'Sign up',
                      textAlign: TextAlign.center,
                      style: TextStyle(
                        color: Color(0xFF5DB074),

```

```

    fontSize: 16,
    fontFamily: 'Inter',
    fontWeight: FontWeight.w600,
    height: 0,
  ),
),
),
],
),
SizedBox(height: 52),
const TextField(
  decoration: InputDecoration(
    hintText: 'Email',
    hintStyle: TextStyle(
      color: Color(0xFFBDBDBD),
      fontSize: 16,
      fontFamily: 'Inter',
      fontWeight: FontWeight.w500,
      height: 0,
    ),
  ),
),
SizedBox(height: 16),
TextField(
  decoration: InputDecoration(
    suffixText: 'Show',
    suffixStyle: TextStyle(
      color: Color(0xFF5DB074),
      fontSize: 16,
      fontFamily: 'Inter',
      fontWeight: FontWeight.w600,
      height: 0,
    ),
    hintText: 'Password',
    hintStyle: TextStyle(
      color: Color(0xFFBDBDBD),
      fontSize: 16,
      fontFamily: 'Inter',
      fontWeight: FontWeight.w500,
      height: 0,
    ),
  ),
),
SizedBox(height: 60),
MainButton(
  text: 'Log in',
  onTap: () {
    Navigator.of(context).push(MaterialPageRoute(builder: (ctx) => TabsScreen()));
  },
),
],
),
),
);
}
}

class MainButton extends StatelessWidget {
  const MainButton({
    super.key,
    required this.text,
    required this.onTap,
    this.color,

```

```

));

final String text;
final VoidCallback onTap;
final Color? color;

@override
Widget build(BuildContext context) {
  return InkWell(
    onTap: onTap,
    child: Container(
      width: double.infinity,
      height: 51,
      padding: const EdgeInsets.symmetric(horizontal: 32, vertical: 16),
      decoration: ShapeDecoration(
        color: color ?? const Color(0xFF5DB074),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(100),
        ),
      ),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.min,
        mainAxisSize: MainAxisSize.min,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text(
            text,
            textAlign: TextAlign.center,
            style: TextStyle(
              color: Colors.white,
              fontSize: 16,
              fontFamily: 'Inter',
              fontWeight: FontWeight.w600,
            ),
            height: 0,
          ),
        ],
      ),
    );
}

```

**ДОДАТОК В**  
**Слайди презентації**

# ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ ЕМОЦІЙ ЗА ЗОБРАЖЕННЯМ

## Дипломна робота

Виконала:  
ст. групи КНТ-128

Дем'янова Дар'я Костянтинівна

Керівник:  
к.т.н, доцент

Т. В. Федорончак

1

Рисунок В.1 – Слайд 1

### Об'єкт дослідження

допомога в розпізнаванні емоцій для людей з розладами аутистичного спектру.

### Предмет дослідження

мобільний додаток для визначення та розпізнавання емоцій у людей з розладами аутистичного спектру.

### Мета роботи

створення зручного та ефективного інструменту для розпізнавання емоцій, спрямованого на полегшення соціальної взаємодії та покращення якості життя людей з розладами аутистичного спектру.

2

Рисунок В.2 – Слайд 2

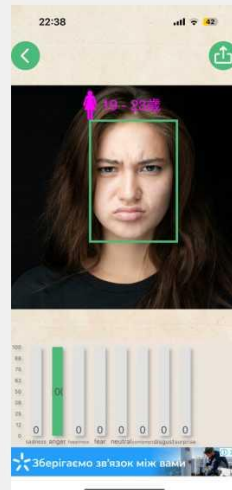
## Порівняння видів нейронних мереж

| Критерій                                            | MLP | CNN | RNN |
|-----------------------------------------------------|-----|-----|-----|
| Здатність до розпізнавання облич та виразів обличчя | +   | ++  | +   |
| Обробка зображень                                   | -   | ++  | -   |
| Відмінності у текстурі та грануляції                | -   | ++  | -   |
| Здатність до аналізу послідовних даних              | -   | -   | +   |
| Табличні набори даних                               | +   | -   | -   |
| Класифікаційні задачі прогнозування                 | +   | +   | +   |
| Задачі регресійного прогнозування                   | +   | +   | +   |
| Швидкість навчання                                  | -   | ++  | ++  |
| Кількість параметрів                                | ++  | ++  | ++  |
| Точність розпізнавання                              | -   | +   | +   |
| Обсяг навчальних даних                              | -   | +   | +   |
| Застосування для аналізу послідовних даних (відео)  | -   | +   | +   |

3

Рисунок В.3 – Слайд 3

## Порівняння аналогів



Критерії порівняння:

- кроссплатформеність
- обробка зображень з різних джерел
- збереження історії фото
- можливість користувача “вгадати” емоцію перед розпізнаванням моделлю
- багатозадачність

4

Рисунок В.4 – Слайд 4

# Побудова Моделі CNN



5

Рисунок В.5 – Слайд 5

## Оптимізація класифікації

Функція втрат – це функція, що характеризує втрати при неправильному прийнятті рішень на основі спостережених даних.

**Cross Entropy Loss** дозволяє ефективно оптимізувати ймовірності приналежності кожного зразка до визначених класів.

Оптимізатор **Adam** дозволяє адаптивно регулювати швидкість навчання для кожної ваги, що полегшує оптимізацію та пришвидшує збіжність моделі.

6

Рисунок В.6 – Слайд 6

# Вибір мови програмування

| Критерій              | Java                                                             | JavaScript                       | Python                                                                                    |
|-----------------------|------------------------------------------------------------------|----------------------------------|-------------------------------------------------------------------------------------------|
| Популярність          | Висока                                                           | Висока                           | Дуже висока                                                                               |
| Екосистема            | Велика, широкий вибір фреймворків та бібліотек                   | Велика, npm для пакетів Node.js  | Велика, використання фреймворків (Django, Flask), багато бібліотек для глибокого навчання |
| Швидкість роботи      | Висока                                                           | Висока                           | Висока                                                                                    |
| Асинхронність         | Зазвичай синхронна, але є асинхронні фреймворки (Spring WebFlux) | Асинхронна (зокрема у Node.js)   | Зазвичай синхронна, але є асинхронні бібліотеки та фреймворки                             |
| Глибоке навчання      | TensorFlow, Deeplearning4j                                       | TensorFlow.js                    | TensorFlow, PyTorch, Keras                                                                |
| Бекенд-розробка       | Для розробки надійних та масштабованих серверних застосунків.    | Більш орієнтована на веброзробку | Веброзробка та серверна розробка                                                          |
| Зручність в навчанні  | Середня                                                          | Легка                            | Легка                                                                                     |
| Підтримка фреймворків | Spring, Hibernate, Apache Struts                                 | Node.js, Nest.js,                | Django, Flask, FastAPI                                                                    |

7

Рисунок В.7 – Слайд 7

## Вибір мови програмування мобільного застосунку



### Flutter

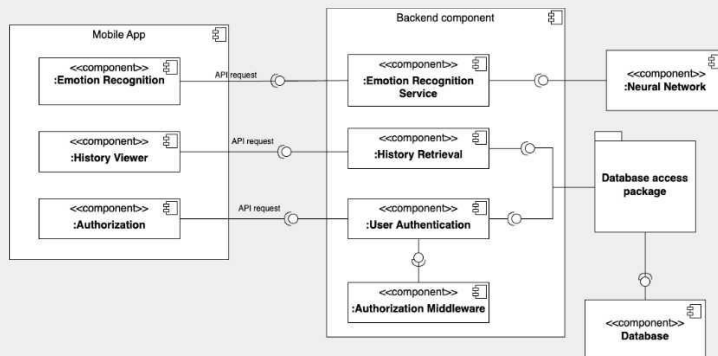
Однаковий код для всіх платформ.  
Зменшений час розробки.  
Швидкість введення на ринок.  
Нативна продуктивність.  
Власний рендеринговий двигун.  
Функція гарячого перевантаження.  
Мова Dart гнучка  
Безпека Flutter, Null-safety  
Документація та підтримка спільноти

Великі та важкі додатки  
Низька популярність Dart.  
Обмежена кількість сторонніх бібліотек.

8

Рисунок В.8 – Слайд 8

## Розробка компонентів системи

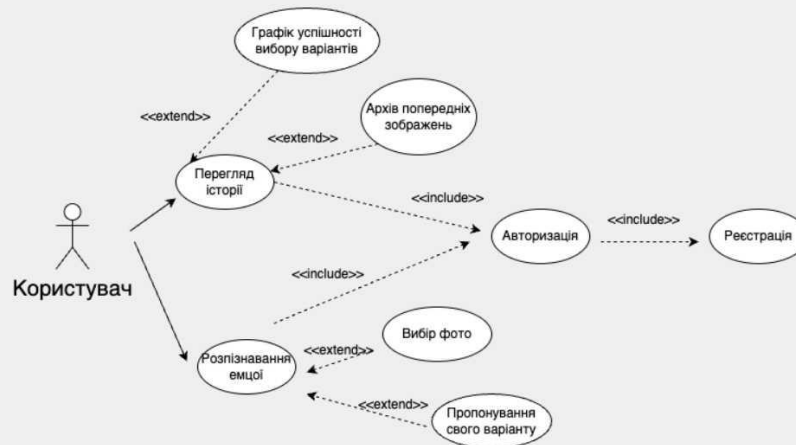


- Мобільний застосунок інтерактивно взаємодіє з бекендом через API. Бекенд взаємодіє з сервісами розпізнавання емоцій, обробки зображень та іншими.
- Сервіс розпізнавання емоцій використовує нейронну мережу та оброблює зображення.
- Сервіс історії обробляє та зберігає історію розпізнавань.
- Модуль авторизації та middleware для авторизації взаємодіють для обробки запитів авторизації та реєстрації.

9

Рисунок В.9 – Слайд 9

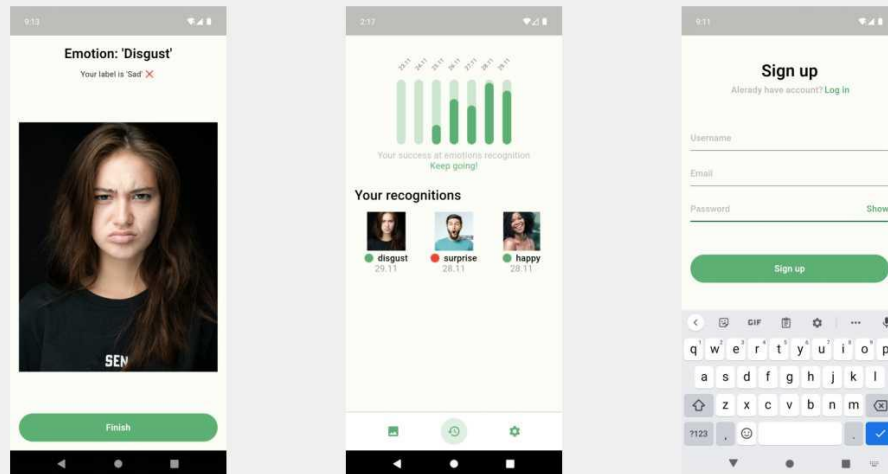
## Діаграма прецедентів



10

Рисунок В.10 – Слайд 10

## Розроблені інтерфейси



11

Рисунок В.11 – Слайд 11

## Файлова структура проекту

```

lib
├── data
│   ├── api_repository.dart
│   ├── image_service.dart
│   └── token_repository.dart
├── models
│   ├── history_dto.dart
│   ├── image_dto.dart
│   ├── label_dto.dart
│   └── user_dto.dart
├── presentation
│   ├── auth
│   │   ├── login_screen.dart
│   │   └── signup_screen.dart
│   ├── main
│   │   ├── emotion_catcher
│   │   │   ├── widgets
│   │   │   │   ├── option.dart
│   │   │   │   ├── emotion_catched.dart
│   │   │   │   ├── emotion_catcher_prepare.dart
│   │   │   └── emotion_choose_label.dart

```

```

├── history
│   ├── widgets
│   │   ├── chart_element.dart
│   │   ├── date_element.dart
│   │   ├── history_chart.dart
│   │   ├── history_element.dart
│   │   ├── history_list.dart
│   │   ├── history_body.dart
│   ├── settings
│   │   ├── settings_screen.dart
│   ├── tabs_screen.dart
├── use_cases
├── auth / domain / cubit
│   ├── auth_cubit.dart
│   ├── auth_state.dart
├── emotions / domain / cubit
│   ├── emotions_cubit.dart
│   ├── emotions_state.dart
├── history / domain / cubit
│   ├── history_cubit.dart
│   ├── history_state.dart
└── main.dart

```

На слайді зображено представлення файлової структури мобільного застосунку

Бекенд-застосунок складався з 4 основних файлів:

- handlers.py
- models.py
- helpers.py
- app.py

12

Рисунок В.12 – Слайд 12

## База даних

MongoDB обрано з кількох вагомих причин:

- гнучкість сховища даних дозволяє адаптувати схему даних без перерви в роботі;
- забезпечує швидкодію завдяки своєму документ-орієнтованому підходу;
- легко масштабується, що дозволяє додавати нові сервери для розподілу навантаження.

| Users    |          |                                      | Predictions   |          |                                                    |
|----------|----------|--------------------------------------|---------------|----------|----------------------------------------------------|
| Поле     | Тип      | Опис                                 | Поле          | Тип      | Опис                                               |
| user_id  | ObjectId | Унікальний ідентифікатор користувача | prediction_id | ObjectId | Унікальний ідентифікатор прогнозу                  |
| username | String   | Ім'я користувача                     | user_id       | ObjectId | Ідентифікатор користувача, пов'язаний із прогнозом |
| email    | String   | Електронна адреса користувача        | user_label    | String   | Мітка користувача                                  |
| password | String   | Хеш пароллю користувача              | image         | Binary   | Зображення, яке використовується для прогнозу      |
|          |          |                                      | date_created  | DateTime | Дата та час створення запису прогнозу              |

13

Рисунок В.13 – Слайд 13

## Висновки

- Були розглянуті існуючі аналоги та методи розпізнавання емоцій
- Розроблено і описано основні компоненти системи, алгоритми програми та структура бази даних.
- Розроблено архітектуру застосунку та інтерфейси
- Було розроблено ПЗ, яке реалізовує усі поставлені задачі та функціональні вимоги. Реалізовано можливість ефективного розпізнавання емоцій

14

Рисунок В.14 – Слайд 14

Дякую за увагу!

15

Рисунок В.15 – Слайд 15