Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет центр післядипломної освіти
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Мобільний класифікатор зображень на основі штучного інтелекту
Mobile Image Classifier Based on Artificial Intelligence
(тема)

Виконав:
студент __ курсу, групи ПЗПІп-22-2
Гавриленко В.В.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. Кириченко І.В
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет центр післядипломної освіти
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Гавриленко Віталію Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Мобільний класифікатор зображень на основі штучного інтелекту

Затверджена наказом по університету від 17.06. 2024р. № 558 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 27.06.2024

3. Вихідні дані до роботи Розробити мобільний додаток під ОС Android для класифікації зображень за темами (люди, тварини, природа) на основі штучного інтелекту.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі та постановка задачі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	20.05.2024	*виконано*
2	Створення специфікації ПЗ	23.05.2024	*виконано*
3	Проектування ПЗ	24.05.2024	*виконано*
4	Розробка ПЗ	28.05.2024	*виконано*
5	Тестування ПЗ	10.06.2024	*виконано*
6	Оформлення пояснювальної записки	11.06.2024	*виконано*
7	Підготовка презентації та доповіді	20.06.2024	*виконано*
8	Попередній захист	15.07.2024	*виконано*
9	Нормоконтроль, рецензування	16.07.2024	*виконано*
10	Здача роботи у електронний архів	19.07.2024	*виконано*
11	Допуск до захисту у зав. кафедри	22.07.2024	*виконано*

Дата видачі завдання 06 травня 2024р.

Студент (ка) _____ Гавриленко В.В.
(підпис)

Керівник роботи _____ доц. Кириченко І.В.

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 85 стор., 43 рис., 12 табл., 21 джерело.

МОБІЛЬНИЙ КЛАСИФІКАТОР ЗОБРАЖЕНЬ, ШТУЧНИЙ ІНТЕЛЕКТ, ANDROID, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, JAVA, COLAB, ШІ, PYTHON, FIREBASE.

Об’єктом роботи є мобільний застосунок під управлінням операційної системи Android, в якому реалізована інтеграція навченої моделі, заснованої на нейронних мережах, яка призначена для класифікації зображень.

Метою розробки є створення мобільного застосунку, який дозволяє користувачам класифікувати зображення за різними темами (люди, тварини, природа) з використанням камери або галереї мобільного пристрою. Важливим аспектом є можливість використання застосунку без необхідності підключення до бази даних та Інтернету, що забезпечує швидкість роботи, легкий доступ та зручність використання додатку для користувачів.

Метод рішення – використання мови програмування Java та фреймворку Spring для розробки застосунку, Google Colab для навчання моделі класифікації зображень, TensorFlow Lite для інтеграції та виконання моделі на мобільному пристрої, інтеграція з файловою системою мобільного пристрою та камерою для отримання та обробки зображень, реєстрації та авторизації користувачів з використанням Firebase Authentication, локалізації застосунку на українську та англійську мови .

В результаті розробки створено мобільний додаток, який здатний класифікувати зображення за темами навченої моделі (люди, тварини, природа), використовуючи нейронні мережі.

MOBILE IMAGE CLASSIFIER, ARTIFICIAL INTELLIGENCE, ANDROID, MACHINE LEARNING, NEURAL NETWORKS, JAVA, COLAB, AI, PYTHON, FIREBASE.

The subject of the work is a mobile application running on the Android operating system, which implements the integration of a trained model based on neural networks designed for image classification.

The goal of the development is to create a mobile application that allows users to classify images into various categories (people, animals, nature) using the camera or gallery of the mobile device. An important aspect is the ability to use the application without the need for a database or Internet connection, ensuring fast performance, easy access, and user convenience.

The solution method involves using the Java programming language and the Spring framework for application development, Google Colab for training the image classification model, TensorFlow Lite for integrating and running the model on the mobile device, integration with the mobile device's file system and camera for obtaining and processing images, user registration and authentication using Firebase Authentication, and localizing the application into Ukrainian and English languages.

As a result of the development, a mobile application has been created that can classify images according to the trained model's categories (people, animals, nature) using neural networks.

Я, Гавриленко Віталій Валерійович, студент гр. ПЗПіп-22-2, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Мобільний класифікатор зображень на основі штучного інтелекту», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

В сучасному світі мобільні технології та штучний інтелект знаходять все більше застосувань у великій кількості сфер, що значно полегшує та розширює можливості щоденного життя людей. Не є виключенням з таких і класифікація зображень, яка використовується для автоматизованого аналізу зображень та їх класифікації з використанням технології штучного інтелекту [4].

Ця кваліфікаційна робота спрямована на розробку мобільного застосунку під управлінням операційної системи Android, який інтегрує навчену модель для класифікації зображень безпосередньо на пристрої користувача. Основною метою дослідження є розгляд можливостей використання нейронних мереж для вирішення завдання класифікації зображень без необхідності постійного з'єднання з мережею Інтернет або використання облачних ресурсів.

Для досягнення поставленої мети у роботі використовуються сучасні засоби програмування та інструменти розробки, зокрема мова програмування Java, фреймворк TensorFlow Lite для мобільних пристроїв, та Google Colab для навчання моделі штучного інтелекту

Ця робота, поєднуючи досліди, перспективи та методи штучного інтелекту з потужностями мобільних пристроїв для забезпечення ефективних рішень в сфері обробки і класифікації зображень, дає зрозуміти актуальність ШІ в повсякденному житті.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної галузі

Мобільні застосунки сьогодні не уявляються без штучного інтелекту (ШІ), що радикально змінює їх взаємодію з користувачами. Одним з напрямків є використання ШІ для класифікації зображень. Мобільні класифікатори зображень дозволяють автоматично розпізнавати об'єкти на фотографіях, забезпечуючи користувачам швидкість та зручність обробки інформації.

Штучний Інтелект – штучні нейронні мережі (AI) є основою алгоритмів глибокого навчання, імітуючи біологічні нейрони людського мозку. Вони складаються з шарів вузлів, які включають вхідний шар, один або кілька прихованих шарів і вихідний шар. Кожен штучний нейрон з'єднується з іншим з відповідною вагою та пороговим значенням. Нейронні мережі залежать від навчальних даних для покращення точності і стають потужними інструментами в інформатиці та штучному інтелекті. Вони можуть ефективно класифікувати та кластеризувати дані, включаючи завдання розпізнавання зображень, що раніше вимагали великих часових затрат [5].

При виконанні кваліфікаційної роботи для навчання моделі обрано платформу Google Colab, яка використовує ресурси Google для обробки даних та навчання з ШІ з використанням TensorFlow та Keras.

Весь процес навчання моделі виконується на мові програмування Python, що є стандартом для роботи з машинним навчанням та ШІ. Python забезпечує широкий набір бібліотек та інструментів для обробки даних, створення та навчання моделей, а також їх оптимізації для різних платформ.

Для навчання моделі необхідно зібрати великий обсяг даних. Зібрані дані (зображення) розподіляються на категорії (люди, тварини, природа) для подальшого використання в навчанні моделі ШІ. Далі відбувається навчання моделі, включаючи інтеграцію у необхідний додаток.

Отже можна сказати, що для розробки та інтеграції застосунків на основі ШІ на мобільних пристроях використовуються різні інструменти та технології. Android є популярною платформою для застосунків, яка підтримує розробку на різних мовах програмування. В поєднанні з мовою програмування Java та фреймворком Spring ці

інструменти дозволяють створювати потужні та гнучкі додатки. TensorFlow Lite забезпечує інтеграцію та виконання навченої моделі на мобільних пристроях. Це легка версія TensorFlow, оптимізована для мобільних і вбудованих пристроїв, що дозволяє виконувати моделі ШІ з високою продуктивністю та низьким споживанням ресурсів [14].

Таким чином, розвиток мобільних застосунків для класифікації зображень на основі штучного інтелекту відкриває нові можливості для користувачів, надаючи їм інструменти для ефективної обробки та аналізу візуальної інформації. Використання передових технологій, таких як TensorFlow Lite, дозволяє забезпечити високу продуктивність та зручність використання цих застосунків на мобільних пристроях.

1.2 Аналіз конкурентів

У сфері мобільних застосунків для класифікації зображень на основі штучного інтелекту існує кілька відомих конкурентів, кожен з яких має свої сильні та слабкі сторони.

Перш ніж порівнювати та проводити детальний та порівняльний аналіз конкурентів, наведемо загальний їх опис:

- Google Lens – це потужний інструмент для розпізнавання об'єктів, розроблений Google. Дозволяє користувачам ідентифікувати об'єкти, перекладати текст, шукати інформацію про продукти та визначати види рослин та тварин.
- Seeing AI від Microsoft – це застосунок, який допомагає людям з вадами зору, використовуючи штучний інтелект для опису об'єктів та сцен.
- CamFind – це мобільний застосунок, який дозволяє користувачам робити фотографії об'єктів і миттєво отримувати інформацію про них.

Нижче в таблиці 1.1 наведено порівняльний аналіз трьох основних конкурентів у цій галузі.

Таблиця 1.1 – Порівняння конкурентів (таблиця виконана самостійно)

Параметр	Google Lens	Microsoft Seeing AI	CamFind
Точність розпізнавання	Висока	Висока	Висока
Інтернет-залежність	Потрібен	Потрібен	Потрібен
Цільова аудиторія	Загальна	Люди з вадами зору	Загальна
Підтримка реального часу	Так	Так	Так
Підтримка багатьох мов	Так	Так	Ні
Конфіденційність даних	Питання конфіденційності	Питання конфіденційності	Питання конфіденційності
Функціональність	Широка	Спеціалізована для допомоги	Універсальна

Аналізуючи дані щодо конкурентів, можна зробити висновок, що жоден не задовольняє всіх функціональних вимог користувача [17].

Перший продукт для аналізу це – Google Lens (рисунок 1.1) та відповідний мобільний додаток (рисунок 1.2).

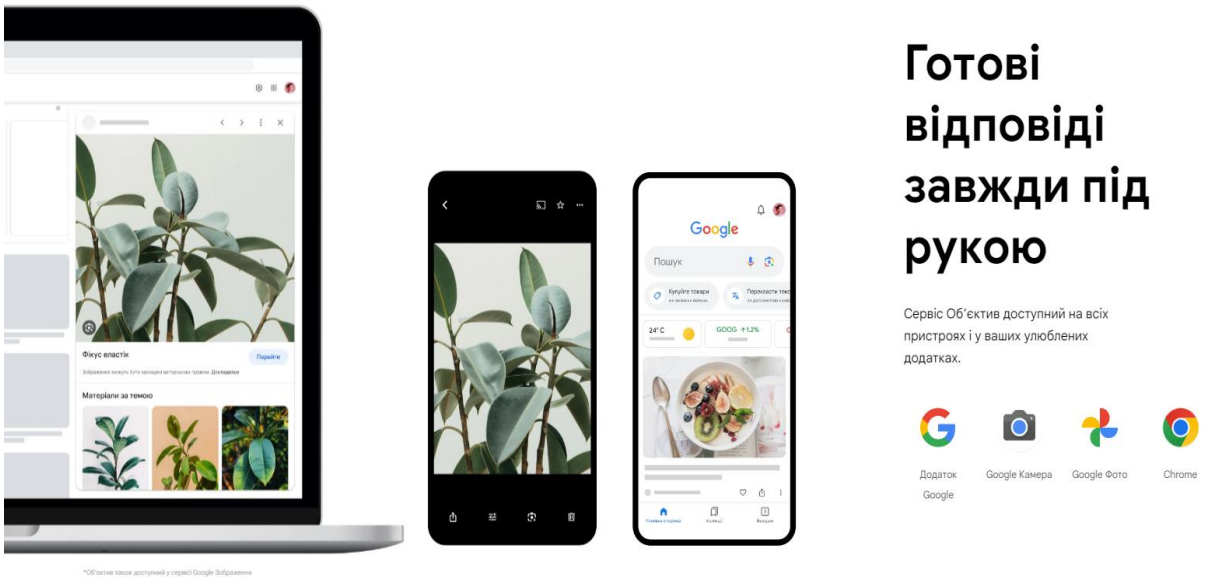


Рисунок 1.1 – Офіційна сторінка ознайомлення з Google Lens (за даними [2])

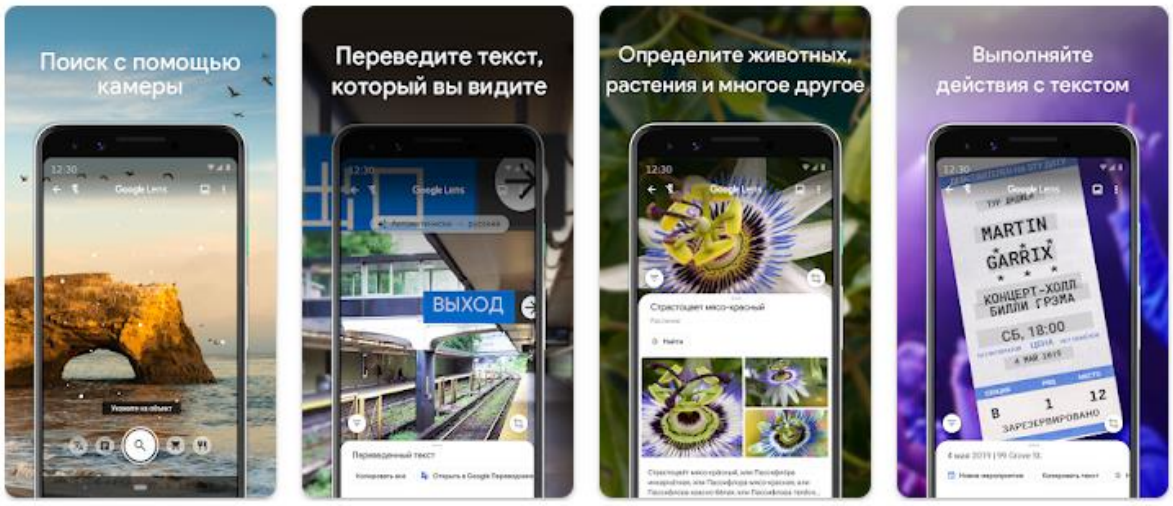


Рисунок 1.2 - Сторінка ознайомлення з додатком Google Lens на play.google.com (за даними [3])

Google Lens є інноваційним інструментом для мобільних пристроїв, що використовується для розпізнавання об'єктів за допомогою камери смартфона. Основні особливості і переваги Google Lens включають:

– розпізнавання об'єктів: Google Lens дозволяє користувачам розпізнавати різні об'єкти, такі як рослини, тварини, предмети одягу, будівлі тощо, просто відсканувавши їх камерою смартфона;
– інтеграція з Google Assistant: функціональність Google Lens є частиною Google Assistant, що робить його доступним для швидкого доступу і використання безпосередньо з асистентом;
– автоматичне визначення інформації: за допомогою Google Lens користувачі можуть автоматично отримувати інформацію про об'єкти, наприклад, опис, цікаві факти, покупні посилання тощо;
– переклад тексту: Google Lens підтримує функцію перекладу тексту, що дозволяє користувачам перекладати текст, який знаходиться на зображеннях, на більше ніж 100 мов;

– інтерактивні можливості: крім розпізнавання, Google Lens пропонує користувачам інтерактивні можливості, такі як створення контактів з візитних карток, додавання подій до календаря тощо;
– покращення продуктивності: використання Google Lens спрощує процес пошуку і отримання інформації про навколишній світ, що значно підвищує обізнаність користувачів.

До недоліків можна віднести залежність від Інтернет-з'єднання, що може бути незручним у місцях з поганим зв'язком. Використання камери для розпізнавання об'єктів може підняти питання щодо приватності користувача, оскільки може збиратися інформація про оточуюче середовище без належного контролю користувача.

Деякі користувачі відзначають, що Google Lens може працювати повільно на старіших або менш потужних пристроях, що обмежує його практичність у деяких сценаріях використання.

Наступний продукт для аналізу – це Microsoft Seeing AI (рисунок 1.3).

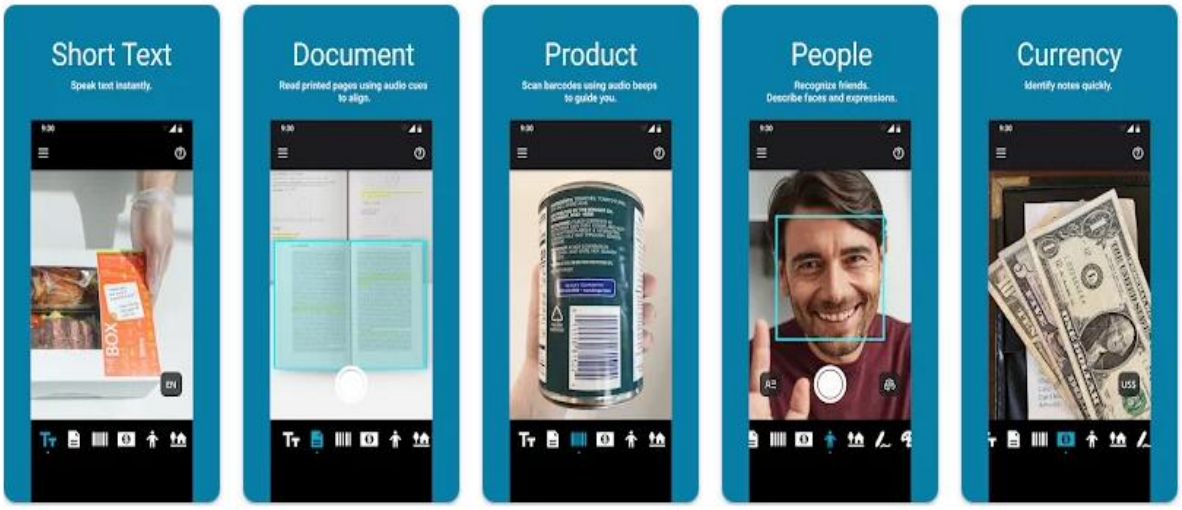


Рисунок 1.3 — Інтерфейс додатку Microsoft Seeing AI (за даними [6])

Microsoft Seeing AI – це ще один значний продукт у сфері розпізнавання образів, який спеціалізується на наданні візуально імперативним людям додаткових засобів спілкування зі світом навколо них. Він розроблений для допомоги людям з

обмеженими можливостями зору та можливість розпізнавання тексту, обличь, об'єктів і навколишнього середовища, що значно полегшує їхнє щоденне життя.

Програма пропонує різні функції, такі як читання тексту, розпізнавання кольорів, орієнтування в просторі, ідентифікація облич, а також озвучування змісту фотографій, що робить її надзвичайно корисним інструментом для щоденного використання.

Microsoft: Seeing AI інтегрується з іншими продуктами Microsoft, що розширює його функціональні можливості і надає додаткові переваги для користувачів.

Використання передових технологій машинного навчання і комп'ютерного зору дозволяє Seeing AI досягати високої точності в розпізнаванні об'єктів і сцен, що важливо для забезпечення коректної роботи програми.

Серед недоліків можна виділити можливість помилок у розпізнаванні об'єктів у складних або поганих умовах освітлення, а також відсутність деяких розширених функцій, які можуть бути доступні в інших комерційних продуктах.

Останнім для порівняння є CamFind (рисунок 1.4).

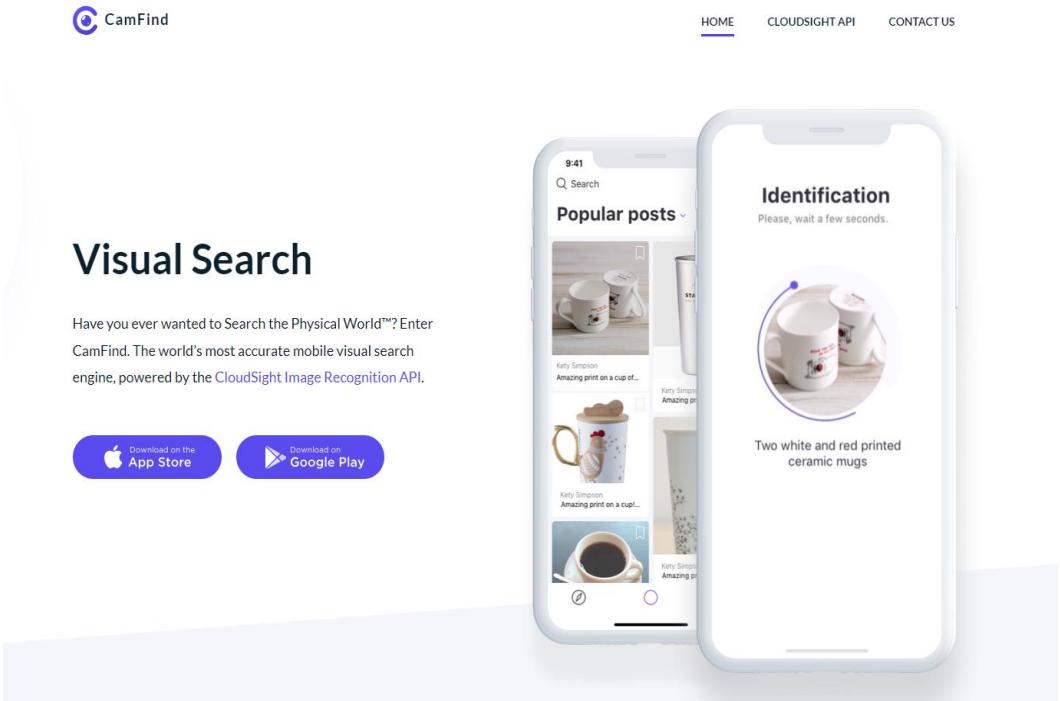


Рисунок 1.4 – Офіційна сторінка CamFind (за даними [1])

Це інноваційний мобільний додаток, який використовує технології комп'ютерного зору для розпізнавання об'єктів і пошуку інформації за допомогою камери смартфону

Основна функція CamFind – це розпізнавання об'єктів на основі фотографій, зроблених користувачем. Додаток використовує передові алгоритми комп'ютерного зору для ідентифікації предметів, розпізнавання тексту на зображеннях і навіть аудіо-розпізнавання. Після розпізнавання об'єкта CamFind надає користувачеві детальну інформацію про цей предмет, включаючи асоційовані з ним товари, послуги або відповідні посилання в Інтернеті. Це дозволяє отримати додаткові знання про будь-який предмет, що був знятий камерою.

Додаток інтегрується з різними онлайн-сервісами і базами даних для надання зрозумілої інформації користувачам швидко і ефективно [18].

Серед недоліків можна відзначити не завжди точне розпізнавання об'єктів, особливо у складних умовах або при наявності багато об’єктних сцен.

1.3 Виявлення проблем та актуалізація рішень

У процесі аналізу та розробки мобільного застосунку для класифікації зображень на основі штучного інтелекту виникла низка проблем та викликів, які потребували вирішення для досягнення оптимальних результатів.

Однією з основних проблем було те, що мобільні пристрої мають обмежені обчислювальні ресурси порівняно з десктопними комп'ютерами. Це створювало виклик щодо ефективного виконання моделей машинного навчання, які потребують значних ресурсів, на мобільних платформах. Як рішення, для навчання моделей було використано Google Colab, що надає доступ до потужних обчислювальних ресурсів. Google Colab забезпечує зручне середовище для написання та виконання коду на Python, пропонує інтеграцію з різними бібліотеками, а також надає безкоштовний доступ до GPU, що суттєво прискорює процес навчання моделей.

Для вирішення проблеми використання ресурсів мобільної платформи при роботі ШІ було використано TensorFlow Lite – це оптимізована версія TensorFlow та призначена для мобільних і вбудованих пристроїв, що забезпечує високу

продуктивність та низьке споживання ресурсів. TensorFlow Lite пропонує ефективні інструменти для конвертації та оптимізації моделей, що робить їх придатними для використання на мобільних пристроях з обмеженими ресурсами [11].

Приймаючи до уваги той факт, що додаток не має у своєму складі бази даних для зберігання зображень та результатів класифікації, було вирішено використовувати файлову систему мобільного пристрою. Це дозволяє швидко отримувати та обробляти дані.

Окремо було приділено увагу питанню безпеки, оскільки фотографії можна віднести до особистої інформації, що вимагає високого рівня безпеки та конфіденційності. У додатку використовуються методи зберігання та обробки зображень у локальній файловій системі мобільного пристрою, що гарантує, що особисті дані користувачів залишаються захищеними, а реєстрація та авторизація користувача забезпечується за допомогою Firebase, що надає простий у використанні SDK та готові бібліотеки користувацького інтерфейсу [7].

1.4 Постановка задачі

Ідея програмного продукту полягає у вирішенні проблем, виявлених у попередньому розділі, шляхом розробки мобільного застосунку для класифікації зображень на основі штучного інтелекту.

Метою роботи є створення ефективного і зручного у використанні мобільного застосунку, який забезпечить високу точність класифікації зображень і надійність обробки даних.

Розроблений програмний продукт повинен надавати такі можливості:

– класифікація зображень за категоріями (люди, тварини, природа) за допомогою камери та галереї мобільного пристрою;
– збереження зображень у відповідних теках на основі результатів класифікації. Наприклад, зображення людей зберігатимуться в теці "люди", зображення тварин – в теці "тварини", а зображення природи – в теці "природа";

– локальне зберігання та обробка зображень у файловій системі мобільного пристрою для забезпечення конфіденційності даних користувачів;
– використання технології TensorFlow Lite для інтеграції та виконання моделей машинного навчання на мобільному пристрої, що забезпечує високу продуктивність та низьке споживання ресурсів;
– забезпечення інтуїтивно зрозумілого інтерфейсу користувача для зручної взаємодії із застосунком.

Програмний продукт має відповідати наступним вимогам:

– безпека – забезпечення високого рівня безпеки та конфіденційності даних користувачів;
– продуктивність – використання оптимізованих моделей машинного навчання для забезпечення високої продуктивності на мобільних пристроях з обмеженими ресурсами;
– зручність використання – розробка інтуїтивно зрозумілого та зручного у використанні інтерфейсу користувача;
– вимоги до ОС та пристрою – підтримка операційної системи Android 8.0 (Oreo) та вище. Пристрій повинен мати мінімум 2 ГБ оперативної пам'яті та чотириядерний процесор для забезпечення належної продуктивності.

Мобільний застосунок має відповідати сучасним стандартам розробки програмного забезпечення і використовувати такі технології та інструменти:

– мова програмування Java для розробки Android-застосунків;
– платформа Google Colab для навчання моделей машинного навчання;
– інструменти для оптимізації моделей TensorFlow Lite для інтеграції та виконання моделей на мобільних пристроях.

Мобільний застосунок виконує роль класифікатора зображень, забезпечуючи високу точність і швидкість обробки, а також надійність і безпеку даних користувачів.

Система повинна використовувати передові технології розробки програмного забезпечення, щоб забезпечити високий рівень продуктивності і задовольнити потреби кінцевих користувачів.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Вимоги до клієнтської частини

В якості клієнтської частини буде виступати мобільний застосунок для системи Android, написаний за допомогою мови програмування Java та Android SDK. До основних функціональних вимог клієнтської частини можна віднести наступні:

– авторизація користувача: користувач може зареєструватись, а вже зареєстрований користувач авторизуватись в застосунку за допомогою авторизаційних даних Google, в разі втрати авторизаційних даних вони можуть бути відновлені за допомогою електронної пошти;
– локалізація додатку: користувач може обирати локалізацію застосунку (англійською або українською мовами);
– класифікація зображень за темами: застосунок буде мати можливість класифікувати зображення за темами (люди, тварини, природа) за допомогою камери та галереї мобільного пристрою;
– збереження зображень у відповідних теках: зображення повинні автоматично зберігатися у відповідних теках на основі результатів класифікації. Наприклад, зображення людей зберігатимуться в теці "люди", зображення тварин – в теці "тварини", а зображення природи – в теці "природа";
– інтуїтивно зрозумілий інтерфейс користувача: розробка зручного у використанні інтерфейсу, який дозволяє легко взаємодіяти із застосунком та швидко отримувати результати класифікації.

В даному переліку виділені основні вимоги до рішення з боку клієнтської частини, без яких дане програмне рішення не матиме сенсу.

2.2 Вимоги до серверної частини

Основна функція серверної частини полягає в обробці зображень, отриманих з камери або галереї мобільного пристрою, та їх класифікації за заданими категоріями (люди, тварини, природа). Після класифікації зображення автоматично зберігаються у відповідних теках на мобільному пристрої. Це рішення забезпечує локальне

зберігання даних, що виключає необхідність передачі зображень через інтернет, тим самим захищаючи конфіденційність користувачів.

Серверна частина застосунку буде включати наступні технології:

– TensorFlow Lite: легка версія TensorFlow, яка спеціально розроблена для мобільних і вбудованих пристроїв. TensorFlow Lite забезпечує високу продуктивність та низьке споживання ресурсів, що дозволяє ефективно виконувати моделі машинного навчання на мобільних пристроях [8];
– Java: мова програмування, яка забезпечує надійність та масштабованість серверної частини. Java є основною мовою для розробки Android-застосунків, що дозволяє інтегрувати серверну частину безпосередньо в мобільний додаток;
– Android SDK: набір інструментів для розробки програмного забезпечення для операційної системи Android. Android SDK забезпечує доступ до функцій камери, файлової системи та інших ресурсів мобільного пристрою.

2.3.Вимоги до інфраструктури

Для забезпечення максимальної відмовостійкості та швидкості роботи мобільного додатку для класифікації зображень було вирішено використовувати нижчеперелічені інструменти та технології.

Модель класифікації використовує TensorFlow Lite для інтеграції та виконання моделей машинного навчання на мобільному пристрої, що забезпечує ефективне виконання моделей навіть на пристроях з обмеженими ресурсами.

Для тренування моделі обрано Google Colab, який дозволяє використовувати потужні GPU для пришвидшення процесу навчання, зменшуючи необхідність у власному потужному обладнанні. Всі зображення, зроблені користувачами або ті, які можуть бути використані додатком, зберігаються безпосередньо на пристрої користувача, що забезпечує швидкий доступ до зображень без необхідності використання інтернету.

Розробка та тестування здійснюються в основному середовищі для розробки мобільного додатку Android Studio, яке дозволяє ефективно створювати та тестувати додаток з підтримкою останніх версій Android.

Ці компоненти разом забезпечують надійність, масштабованість та ефективність роботи мобільного додатку для класифікації зображень, а також полегшують процес його розробки, тестування та підтримки.

3 АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Макети інтерфейсу користувача

Макети інтерфейсу користувача були розроблені за допомогою онлайн-платформи Figma [9], що дозволило створити високоякісні візуальні представлення інтерфейсу мобільного додатку класифікатора зображень.

Нижче представлено опис трьох ключових макетів, які ілюструють різні стадії взаємодії користувача з додатком.

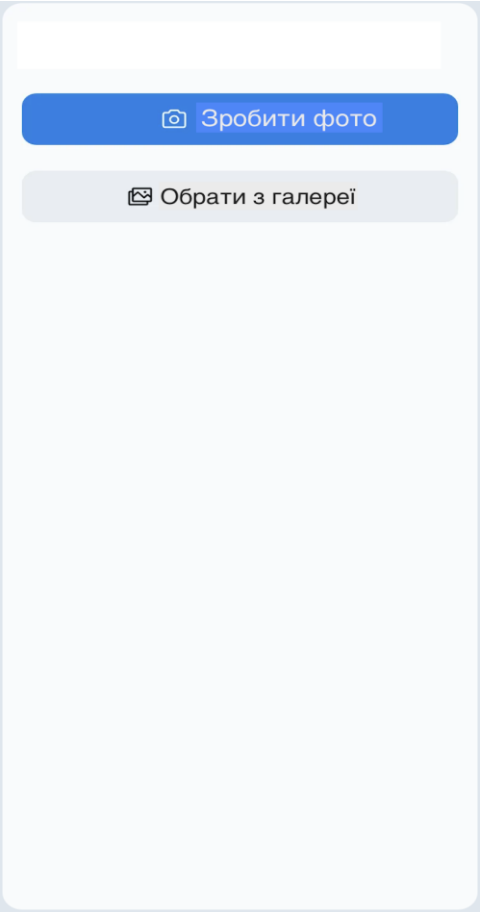


Рисунок 3.1 Макет головної сторінки (виконано самостійно)

Головна сторінка (Рисунок 3.1) додатку надає користувачам два основних варіанти взаємодії: зробити нове фото за допомогою камери телефону або вибрати існуюче зображення з файлової системи. Це забезпечує легкий доступ до основних функцій додатку і стимулює користувача до активного використання різних можливостей додатку.

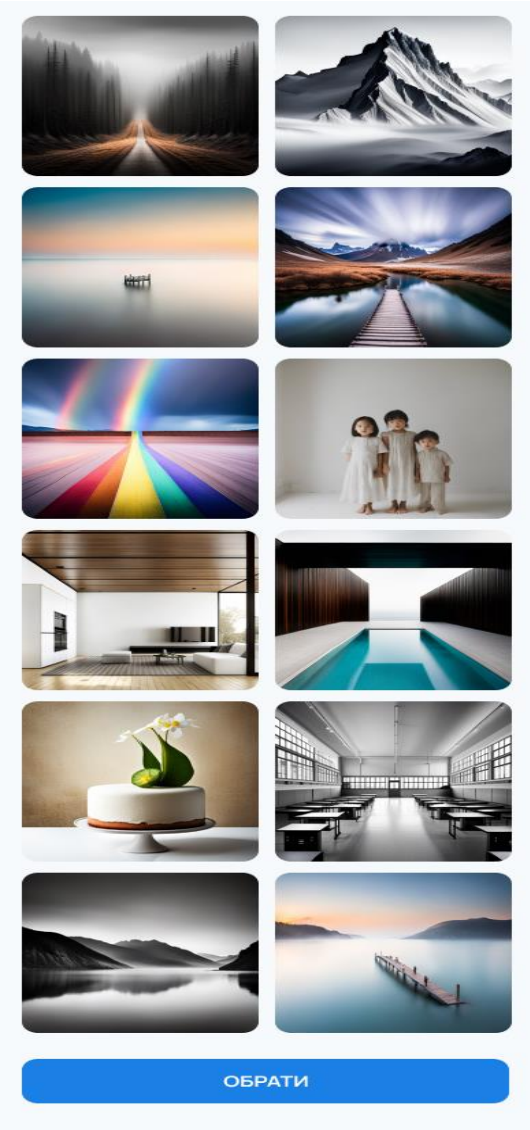


Рисунок 3.2 Макет вибору зображення з галереї (виконано самостійно)

На рисунку 3.2 продемонстровано відкриту галерею, де користувач може переглядати всі доступні фотографії та вибирати одну для подальшої класифікації. Вибір зображень відбувається через просте тапання по зображенню, що забезпечує швидку взаємодію.

Галерея має сучасний вигляд з ефективним розміщенням зображень, яке максимізує використання екранного простору і дозволяє користувачам легко орієнтуватися між фотографіями.

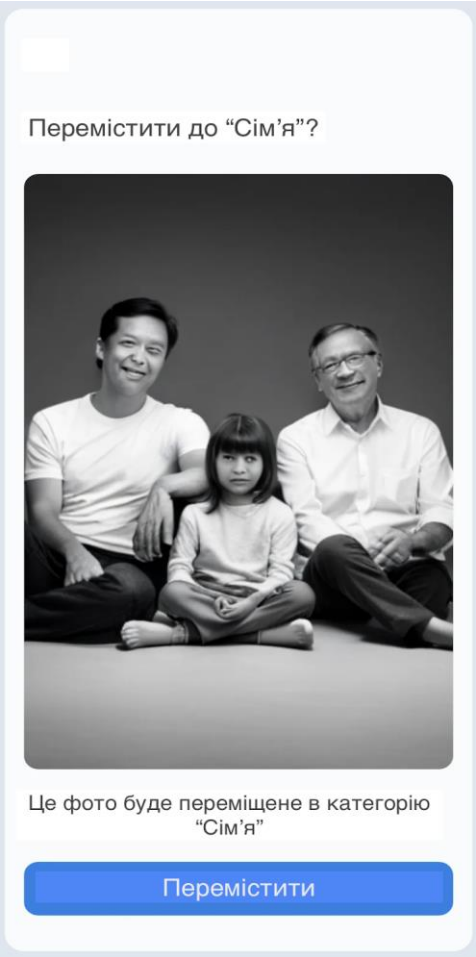


Рисунок 3.3 Макет результату класифікації (виконано самостійно)

На екрані результату класифікації (Рисунок 3.3) користувач бачить результат класифікації зображення. Після класифікації, зображення можна перевірити та зберегти у відповідну категорію. Користувач має можливість вибрати "Перемістити", щоб перенести фото в спеціально створену папку для кожної категорії, наприклад "Сім’я".

Інтерфейс представляє собою чистий та зрозумілий дизайн, з чіткими вказівками та легкодоступними кнопками. Приклади розробленого інтерфейсу наведено на малюнках В.1 – В.5 додатку В.

3.2 Діаграма станів

Діаграма станів в програмному забезпеченні важлива для візуалізації поведінки системи або компонентів системи у відповідь на зовнішні події чи умови [9]. Вона

демонструє різні стани додатку та переходи між ними, що допомагає зрозуміти, як додаток реагує на дії користувача та як взаємодіє з іншими компонентами системи.

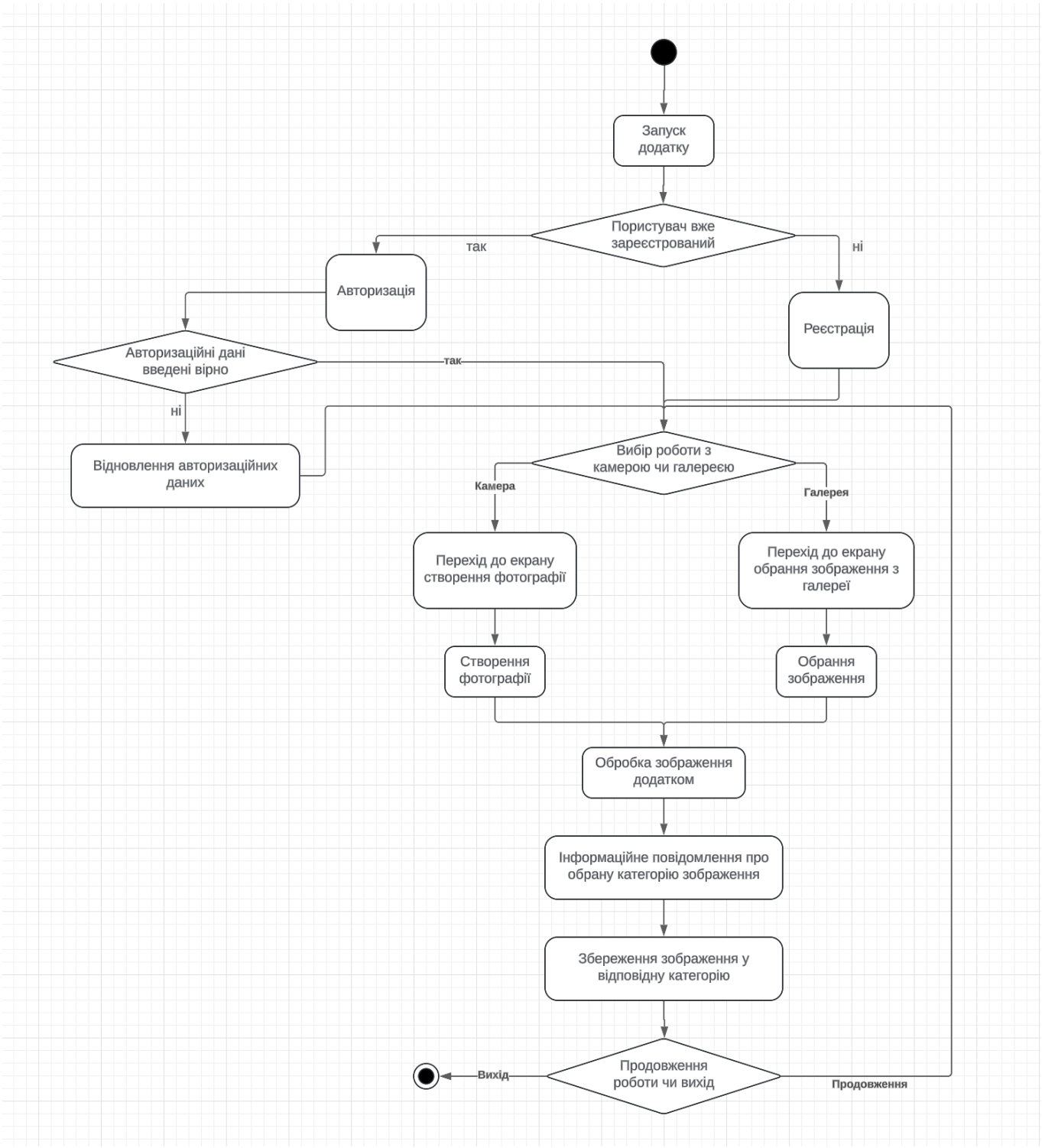


Рисунок 3.4 – Діаграма станів (рисунок виконаний самостійно)

На рисунку 3.4 наведено діаграму станів розробленого мобільного класифікатора зображень.

Початковий стан додатку представлений чорною точкою. Це точка, з якої починається робота додатку після його відкриття. Перший стан, в якому перебуває додаток після запуску, називається “Відкриття додатку”. Після відкриття додаток переходить до стану “Головний екран”. З головного екрану користувач може перейти до реєстрації або авторизації, якщо ще не зареєстрований.

Стан “Реєстрація або Авторизація” охоплює процес введення даних користувача для реєстрації або авторизації. Цей стан має підстани, такі як “Введення даних”, “Успішна авторизація” та “Помилка авторизації”. Після успішної авторизації додаток повертається до стану “Головний екран”.

Далі користувач переходить до стану “Вибір джерела”, де він обирає, звідки буде завантажувати зображення – з камери чи з галереї. Після цього додаток переходить до стану “Вибір зображення”, де користувач може зробити знімок або вибрати зображення з галереї. Вибране зображення завантажується, і додаток переходить до стану “Завантаження зображення”.

Після завантаження зображення додаток переходить до стану “Відправка на класифікацію”, де зображення відправляється на сервер для обробки. Сервер, використовуючи TensorFlow Lite, обробляє зображення, і додаток переходить до стану “Очікування результату”. Після завершення класифікації додаток отримує результат і переходить до стану “Отримання результату”. Потім додаток зберігає класифіковане зображення у відповідну теку в стані “Збереження зображення”.

Коли зображення збережено, додаток переходить до стану “Показ результату”, де користувач може побачити результат класифікації. Після цього користувач може зробити новий запит, повернувшись до стану “Вибір джерела”. Кінцевий стан додатку представлений чорною точкою з кільцем, що показує завершення роботи додатку або поточної сесії.

Діаграма станів є цінним інструментом для планування та проектування логіки додатку, забезпечуючи зрозумілий опис поведінки системи. Вона також дозволяє виявити можливі логічні помилки або недоліки в роботі додатку на ранніх етапах розробки.

3.3 Діаграма компонентів

Діаграма компонентів показує, як різні компоненти системи пов’язані між собою та які функції вони виконують [9].

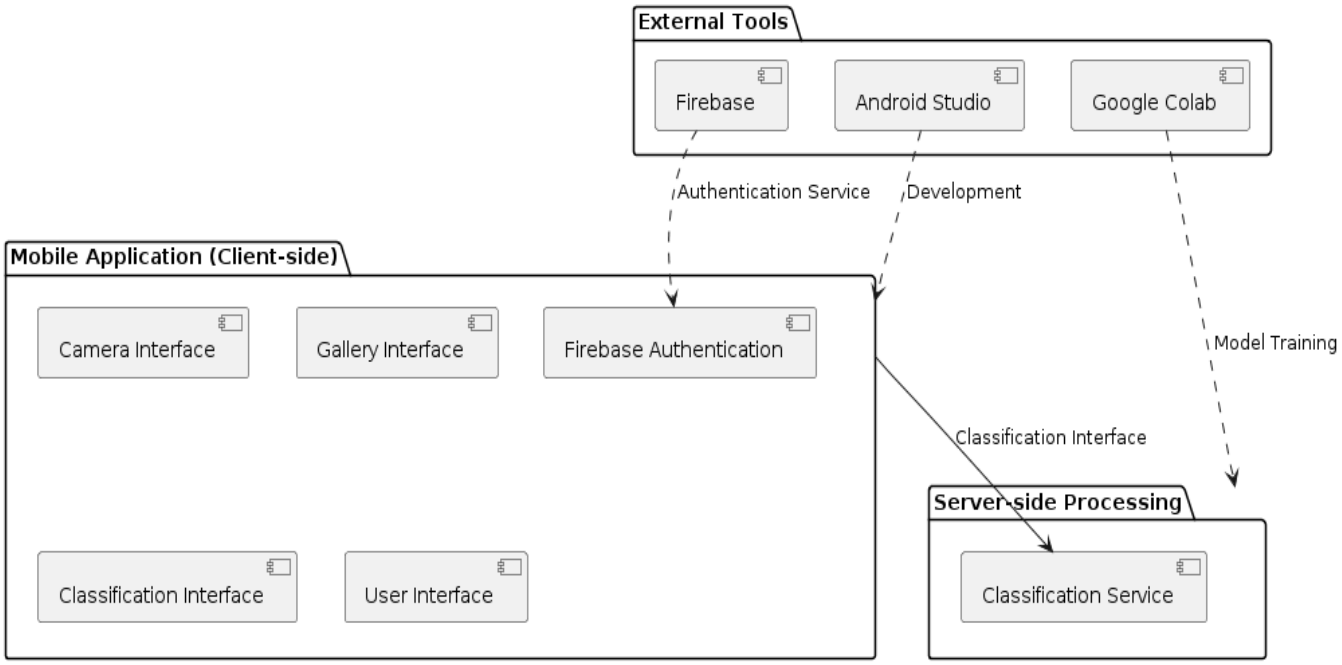


Рисунок 3.5 – Діаграма компонентів (рисунок виконаний самостійно)

Як видно з рисунку 3.5 (діаграма компонентів розробленого мобільного додатку), мобільний додаток складається з кількох основних компонентів, що працюють на клієнтській стороні. Ці компоненти включають інтерфейси камери (Camera Interface), галереї (Gallery Interface), класифікації (Classification Interface) та користувача (User Interface). Додаток також використовує Firebase Authentication для реєстрації та авторизації користувачів, що забезпечує безпечний доступ до системи.

Серверна частина системи відповідає за обробку зображень та їх класифікацію. Вона складається з Classification Service. Серверна частина взаємодіє з TensorFlow Lite, що забезпечує машинне навчання та класифікацію зображень.

Для розробки та тестування мобільного додатку використовуються кілька зовнішніх інструментів. Google Colab використовується для тренування моделей машинного навчання, що дозволяє ефективно навчати моделі з використанням потужних GPU. Android Studio є основним середовищем для розробки та тестування додатку, забезпечуючи підтримку останніх версій Android.

Діаграма компонентів показує, як мобільний додаток взаємодіє з серверною частиною через інтерфейс класифікації. Серверна частина використовує TensorFlow Lite для обробки зображень та повертає результати класифікації назад до мобільного додатку. Firebase забезпечує реєстрацію та авторизацію користувачів, дозволяючи їм безпечно отримувати доступ до додатку та його функцій. Google Colab інтегрується з серверною частиною для тренування моделей.

Діаграма компонентів допомагає зрозуміти архітектуру системи та взаємодію між її частинами. Вона є важливим інструментом для планування та проектування системи, забезпечуючи чітке уявлення про те, як різні компоненти пов’язані між собою та які функції вони виконують.

3.4 Діаграма активностей

Діаграма активностей є важливим інструментом для візуалізації робочих процесів і логіки поведінки мобільного додатку. Вона показує послідовність дій і рішень, що приймаються під час виконання певних задач, та допомагає зрозуміти, як додаток реагує на дії користувача [8].

Діаграма активностей для розробленого мобільного додатку (рисунок 3.6) починається з дії користувача, який відкриває додаток. Після відкриття додатку користувач переходить до стану реєстрації або авторизації за допомогою Firebase. Цей процес включає введення даних користувача та їх перевірку на сервері Firebase. Якщо дані вірні, користувач успішно авторизується і переходить до головного екрану додатку. У разі помилки аутентифікації користувачеві пропонується повторити спробу.

Після успішної авторизації користувач переходить до вибору джерела зображення, яке може бути камерою або галереєю. Вибравши джерело, користувач робить знімок або вибирає зображення з галереї. Зображення завантажується в мобільний додаток, після чого відправляється на серверну частину для класифікації.

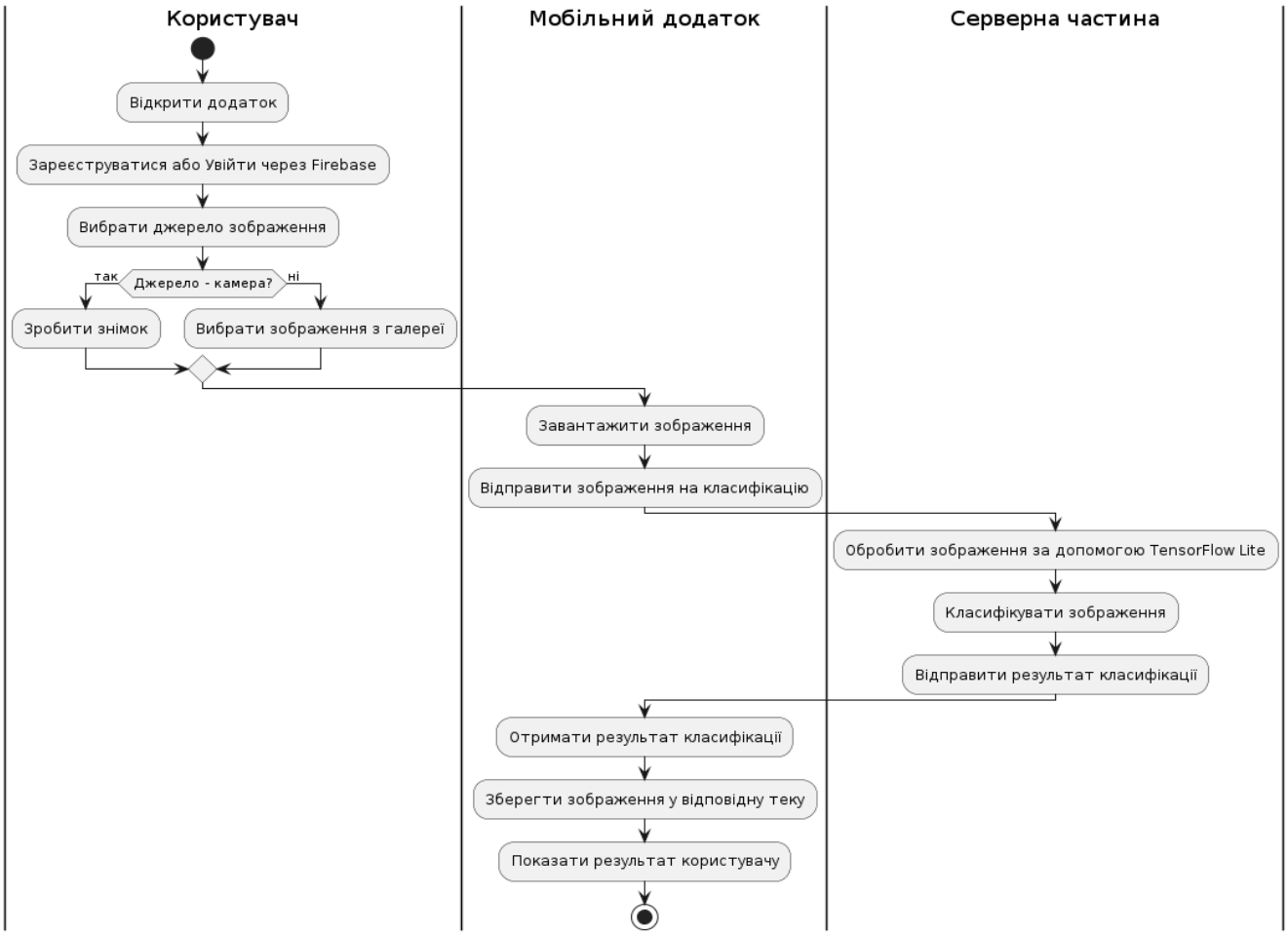


Рисунок 3.6 – Діаграма активностей (рисунок виконаний самостійно)

Серверна частина, використовуючи TensorFlow Lite, обробляє зображення і виконує його класифікацію. Під час цього процесу додаток перебуває в стані очікування результату. Після завершення класифікації серверна частина надсилає результат назад до мобільного додатку. Мобільний додаток отримує результат класифікації, зберігає зображення у відповідну теку (наприклад, “люди”, “тварини”, “природа”) і показує результат користувачеві.

Після того як зображення збережено та результат класифікації показано користувачеві, додаток повертається до головного екрану, де користувач може зробити новий запит, вибравши нове джерело зображення або вийти з додатку.

Діаграма активностей допомагає візуалізувати логіку роботи додатку та порядок виконання дій. Вона є важливим інструментом для планування та проектування додатку, оскільки забезпечує чітке розуміння послідовності дій і рішень,

які приймаються в системі. Діаграма активностей також дозволяє виявити можливі логічні помилки або недоліки в роботі додатку на ранніх етапах розробки.

3.5 Діаграма послідовності

Діаграма послідовності реалізована для візуалізації послідовності взаємодій між об'єктами, що дозволяє відобразити взаємодію об'єктів між собою та послідовність взаємодії [8].

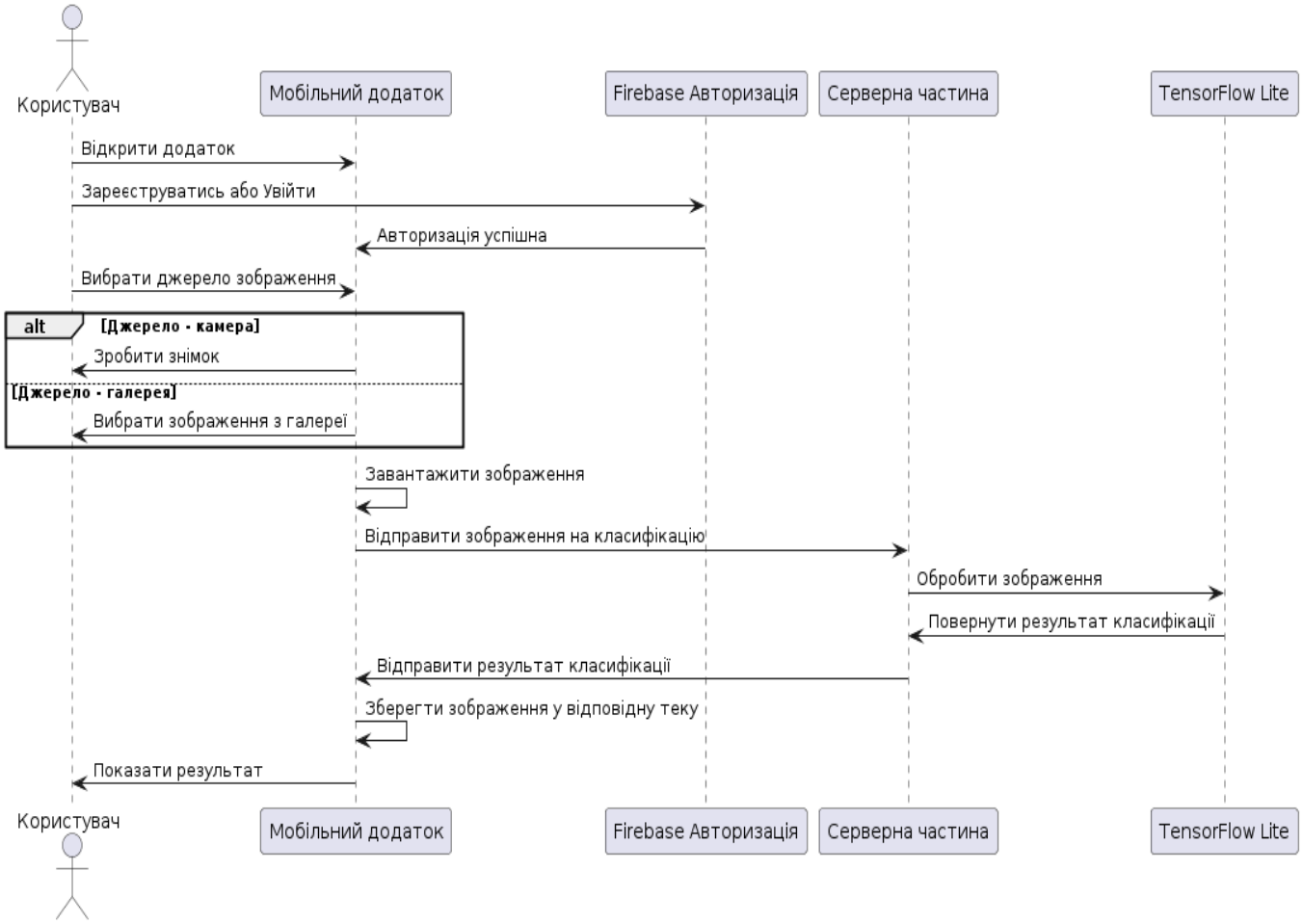


Рисунок 3.7 – Діаграма послідовності (рисунок виконаний самостійно)

Діаграма послідовності для розробленого мобільного додатку (рисунок 3.7) відображає взаємодію між користувачем, мобільним додатком, Firebase для аутентифікації, серверною частиною та TensorFlow Lite під час процесу класифікації зображень.

Спочатку користувач відкриває додаток і проходить процедуру реєстрації або авторизації за допомогою Firebase. Цей процес включає відправку даних користувача

до Firebase для перевірки та отримання підтвердження успішної аутентифікації, після чого користувач отримує доступ до функцій додатку.

Після успішної авторизації користувач вибирає джерело зображення, яке може бути камерою або галереєю. Якщо користувач вибирає камеру, він робить знімок; якщо галерею – вибирає зображення з наявних. Мобільний додаток завантажує вибране зображення і відправляє його на серверну частину для класифікації. Серверна частина, у свою чергу, передає зображення до TensorFlow Lite для обробки. TensorFlow Lite проводить класифікацію зображення та повертає результат до серверної частини.

Серверна частина надсилає результат класифікації назад до мобільного додатку. Мобільний додаток отримує результат, зберігає класифіковане зображення у відповідну теку і показує результат користувачеві. Таким чином, діаграма послідовності показує весь процес від взаємодії користувача з додатком до отримання та відображення результату класифікації.

Ця діаграма допомагає зрозуміти порядок дій і взаємодію між різними компонентами системи під час виконання конкретної задачі. Вона є корисним інструментом для планування та проектування системи, оскільки дозволяє чітко побачити, як різні частини системи пов’язані між собою і як вони взаємодіють під час роботи.

В цілому, діаграма послідовності для розробленого мобільного додатку з використанням Firebase для аутентифікації та TensorFlow Lite для класифікації зображень забезпечує чітке уявлення про те, як система реагує на дії користувача і які кроки виконує для обробки та відображення результатів класифікації. Це допомагає забезпечити ефективну розробку та підтримку додатку, а також забезпечує зрозумілу та детальну документацію процесів, що відбуваються в системі.

3.6 Проектування бази даних

У контексті розробки мобільного додатку класифікатора зображень на основі штучного інтелекту важливим є рішення щодо використання або невикористання бази даних. В даному додатку база даних не реалізована, і таке рішення є доцільним

з кількох причин. По-перше, незбереження стану між сесіями та локальне зберігання даних дозволяють додатку виконувати класифікацію зображень без необхідності зберігати інформацію про попередні сесії або результати класифікації, обробляючи і виводячи всі дані в реальному часі без збереження результатів у довгострокову пам’ять. Це забезпечує простоту та швидкість роботи додатку, оскільки відсутність бази даних уникає зайвої складності, що може виникнути при її інтеграції та управлінні, спрощуючи архітектуру та підтримку додатку. Відсутність операцій з читанням/записом до бази даних забезпечує більш швидкий відгук додатку, оскільки всі операції з даними виконуються безпосередньо в пам’яті пристрою.

Крім того, невикористання бази даних знижує ризики, пов’язані з зберіганням особистих даних користувачів, таких як зображення, що є особливо важливим при обробці особистих чи конфіденційних зображень, які можуть містити відомості чутливого характеру. У мобільному контексті використання, де мобільні пристрої мають обмежені обчислювальні ресурси та обмеження пам’яті, відсутність бази даних дозволяє зекономити цінні системні ресурси для основних операцій, пов’язаних з класифікацією зображень.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

Операційна система Android була обрана як основна платформа для розробки мобільного класифікатора зображень завдяки її популярності, широкому розповсюдженню та розвиненій екосистемі інструментів для розробки. Android підтримується великою кількістю пристроїв різних виробників, що забезпечує доступність додатка для широкої аудиторії. Крім того, Android SDK (Software Development Kit) надає всі необхідні інструменти для розробки, тестування та розгортання мобільних додатків.

Мовою програмування була обрана Java – як основна мова програмування для розробки додатка через її стабільність, продуктивність та популярність серед розробників Android-додатків. Вона забезпечує надійну платформу для створення складних додатків з багатим функціоналом. Велика кількість бібліотек і фреймворків для Java також сприяє швидкому та ефективному розвитку проекту [16].

Для реалізації функції авторизації та аутентифікації користувачів використовується Firebase Authentication. Цей сервіс дозволяє легко інтегрувати систему реєстрації та входу користувачів, забезпечуючи високий рівень безпеки та зручність використання. Firebase надає простий і безпечний спосіб управління користувацькими акаунтами, підтримуючи різні методи авторизації, включаючи електронну пошту, телефонні номери та соціальні мережі (рисунок 4.1)

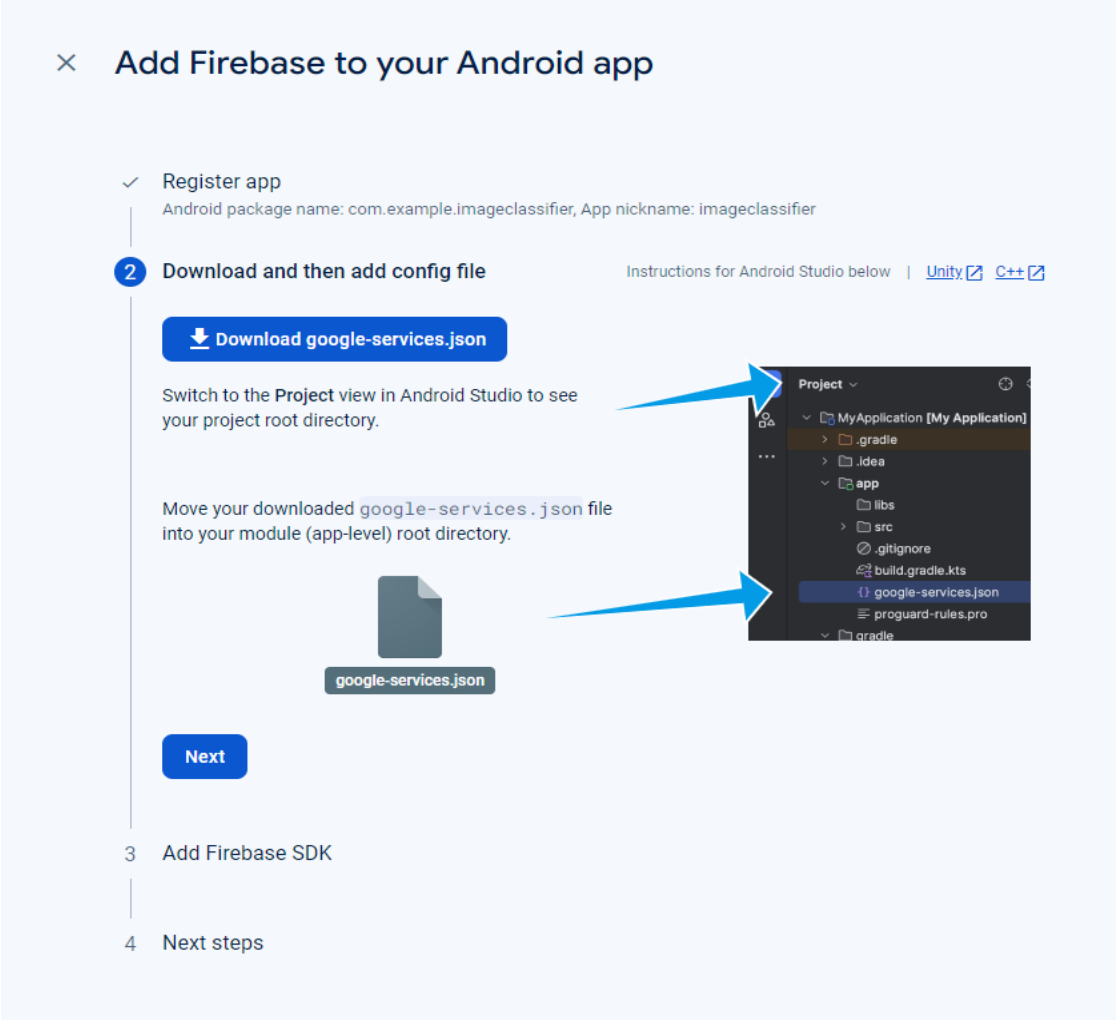


Рисунок 4.1 – Інтеграція Firebase Authentication (виконано самостійно)

У даному проекті відсутня необхідність у використанні бази даних для зберігання інформації. Вся необхідна для роботи додатка інформація обробляється та зберігається локально на мобільному пристрої користувача. Це дозволяє уникнути додаткових витрат на розгортання та підтримку баз даних, а також забезпечує підвищену безпеку та конфіденційність даних користувачів, оскільки вони не передаються через мережу.

Процес навчання моделі класифікації зображень здійснюється за допомогою платформи Google Colab. Google Colab надає доступ до потужних обчислювальних ресурсів Google, що дозволяє ефективно обробляти великі набори даних та навчати моделі глибокого навчання [15]. Google Colab спрощує процес налаштування середовища для машинного навчання, оскільки всі необхідні бібліотеки та інструменти вже встановлені та готові до використання (рисунок 4.2).

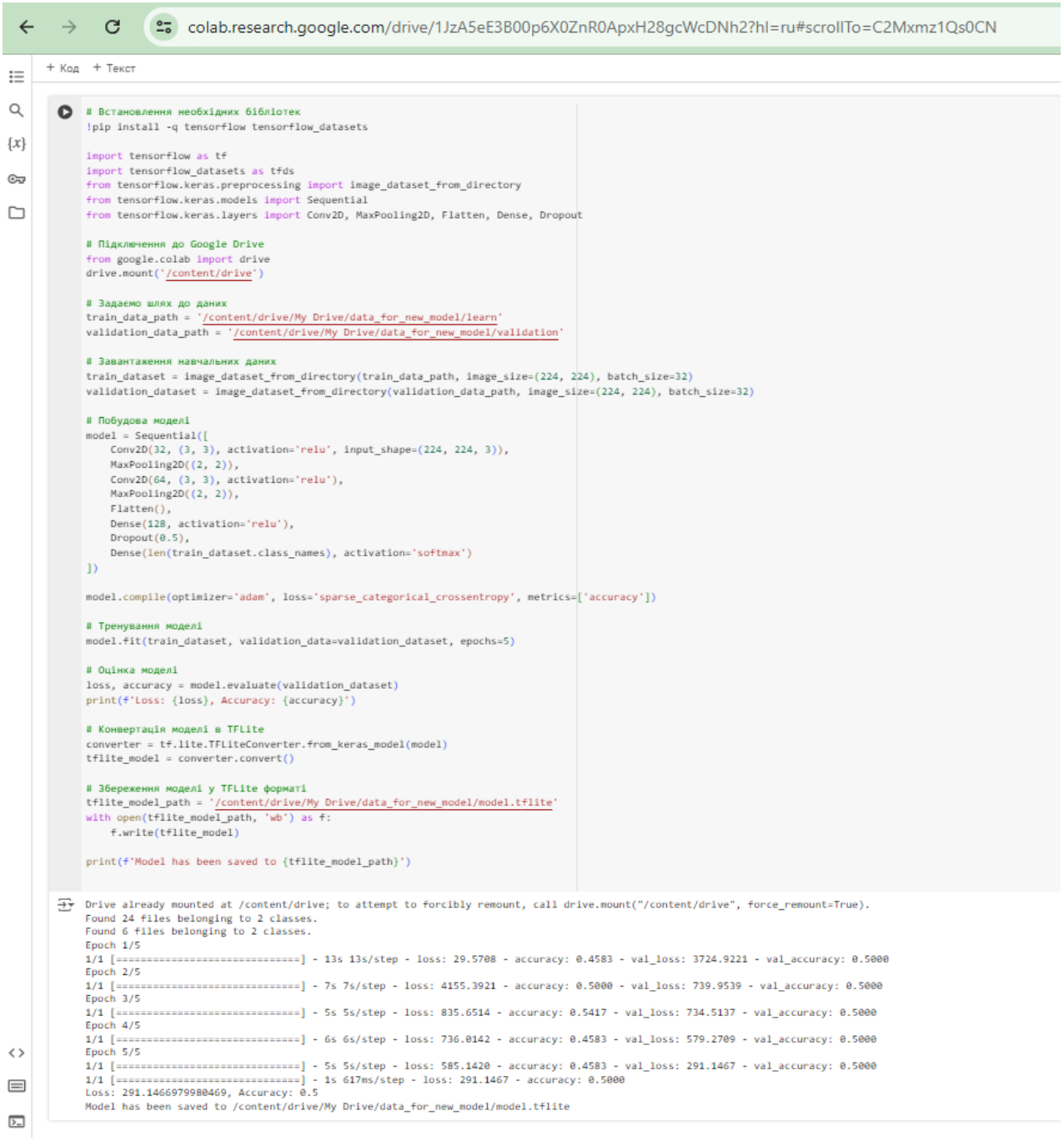


Рисунок 4.2 – Навчання моделі на Colab (виконано самостійно) [21]

Для інтеграції та виконання навченої моделі на мобільному пристрої використовується TensorFlow Lite (на рисунку 4.3 представлено підключення відповідних залежностей до проекту). TensorFlow Lite забезпечує оптимізацію моделей машинного навчання для роботи на пристроях з обмеженими ресурсами, зменшуючи їх розмір та підвищуючи продуктивність [7]. Це дозволяє забезпечити швидке та ефективне виконання моделей класифікації зображень на мобільних пристроях, навіть якщо вони мають обмежені апаратні можливості.

```
implementation 'org.tensorflow:tensorflow-lite:2.8.0'
implementation 'org.tensorflow:tensorflow-lite-gpu:2.8.0'
implementation 'org.tensorflow:tensorflow-lite-support:0.3.1'
implementation 'org.tensorflow:tensor'
```

Рисунок 4.3 – Залежності для TensorFlow Lite (виконано самостійно)

Весь процес навчання моделі виконується на мові програмування Python. Python є стандартом у сфері машинного навчання та штучного інтелекту завдяки своїй простоті та великій кількості доступних бібліотек, таких як TensorFlow та Keras. Використання Python дозволяє швидко розробляти та тестувати моделі машинного навчання, а також легко інтегрувати їх у мобільний додаток за допомогою TensorFlow Lite [15].

На рисунку 4.4 наведено приклад використання моделі.

```
private void classifyImage(Bitmap bitmap) {
    if (bitmap != null) {
        bitmap = Bitmap.createScaledBitmap(bitmap, 224, 224, false);
        try {
            Model model = Model.newInstance(getApplicationContext());

            TensorImage tensorImage = new TensorImage(DataType.FLOAT32);
            tensorImage.load(bitmap);

            Model.Outputs outputs = model.process(tensorImage.getTensorBuffer());
            TensorBuffer outputBuffer = outputs.getOutputFeature0AsTensorBuffer();

            float[] confidences = outputBuffer.getFloatArray();
            int maxIndex = 0;
            float maxConfidence = 0;
            for (int i = 0; i < confidences.length; i++) {
                if (confidences[i] > maxConfidence) {
                    maxConfidence = confidences[i];
                    maxIndex = i;
                }
            }

            String[] classes = {"People", "Landscapes", "Animals"};
            currentCategory = classes[maxIndex];
            String result = "Результат класифікації: " + currentCategory;
            classificationResult.setText(result);

            model.close();
        } catch (IOException e) {
            e.printStackTrace();
            Toast.makeText(this, "Помилка при обробці зображення", Toast.LENGTH_LONG).show();
        }
    } else {
        Toast.makeText(this, "Зображення не вдалося завантажити", Toast.LENGTH_LONG).show();
    }
}
```

Рисунок 4.4 – Використання TensorFlow Lite (виконано самостійно)

Мобільний класифікатор зображень інтегрується з файловою системою мобільного пристрою для отримання та збереження зображень. Це дозволяє користувачам завантажувати зображення з галереї або робити нові знімки за допомогою камери для подальшої класифікації. Інтеграція з файловою системою забезпечує зручність використання додатка та дозволяє зберігати результати класифікації локально на пристрої користувача.

Для автоматизації збірки проекту обрано Gradle.

Gradle – це система автоматизації збирання, яка має вирішальне значення для розробки Android, поєднуючи гнучкість із надійною функціональністю. Є сучасним інструментом, що забезпечує ефективне управління залежностями, гнучку конфігурацію збірок та підтримку масштабованості великих проектів. Gradle.

Офіційна підтримка Android: Оскільки Android Studio (офіційна IDE для Android) використовує Gradle, це фактичний стандарт для проектів Android.

Основні можливості Gradle:

- управління залежностями: Gradle дозволяє легко додавати і оновлювати бібліотеки та інші залежності проекту. Для Android проектів це означає легке підключення сторонніх бібліотек, таких як бібліотеки Firebase, TensorFlow Lite та інші;
- гнучка конфігурація збірок: Gradle пропонує високу гнучкість у налаштуванні процесів збірки. Це включає можливість налаштування різних типів збірок (debug, release), конфігурацію мінімальних та цільових версій SDK;
- масштабованість: Gradle добре підходить для великих проектів з великою кількістю модулів та залежностей [10].

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 30
    defaultConfig {
        applicationId "com.example.myapp"
        minSdkVersion 16
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}
dependencies {
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.3.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}
```

Рисунок 4.5 – Приклад інтеграції Gradle (виконано самостійно)

На рисунку 4.5 представлено приклад інтеграції та налаштування Gradle в розробленому програмному додатку.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування програмного забезпечення є одним з ключових етапів розробки, оскільки дозволяє виявити дефекти та помилки на ранніх стадіях, забезпечуючи тим самим вищу якість кінцевого продукту, а також допомагає зрозуміти вимоги до системи ще на етапі розробки програмного продукту. В процесі розробки і тестування також важливим є використання інструментів, які б дозволяли відстежувати та керувати помилками, дефектами, завданнями та іншими аспектами проекту. Одними з них є JIRA, Bugzilla, Trello, але кожен сам вирішує чим буде користуватись [12]. В даному випадку це xls файли, які містять інформацію для перевірок. Так як оновлення додатку не буде частим, цього буде достатньо для аналізу тестування та відстеження змін [18].

В процесі розробки та тестування були виділені та застосовані нижчеперелічені види.

Функціональне тестування (Functional Testing): Перевірка виконання основних функцій додатку, таких як реєстрація, авторизація, зйомка за допомогою камери пристрою, завантаження фото на обробку з файлової системи додатку, обробка та класифікація зображень, з аналізом відображених результатів.

Тестування інтерфейсу користувача (UI Testing): Оцінка коректності роботи та зручності користувацького інтерфейсу, включаючи навігацію, розташування елементів та відповідність дизайну.

Тестування зручності використання (Usability Testing): Оцінка інтуїтивної зрозумілості та зручності використання додатку, включаючи логіку взаємодії та задоволеність користувачів.

Тестування інсталяції (Installation Testing): Перевірка процесу встановлення додатку на різних пристроях і під різними умовами, включаючи оновлення та видалення додатку [20].

Для виконання функціонального тестування, а також тестування інсталяції було сформовано чек-лист та написані тест кейси.

Зміст розробленого чек-листа представлено в таблиці 5.1.

Таблиця 5.1 — чек-лист (виконана самостійно)

№	Назва тесту	Виконано (Так/Ні)
1	Інсталяція додатку	так
2	Запуск додатку після встановлення	так
3	Видалення додатку	так
4	Зйомка зображення за допомогою камери	так
5	Збереження знімку у додатку	так
6	Завантаження зображення з галереї	так
7	Відображення завантаженого зображення	так
8	Виконання класифікації зображень	так
9	Відображення результатів класифікації	так
10	Класифікація зображень в категорії 'люди' або 'собаки'	так
11	Відображення елементів інтерфейсу на всіх екранах	так
12	Працездатність кнопок та інтерактивних елементів	так
13	Коректна навігація між екранами	так
14	Зручність та інтуїтивність інтерфейсу	так
15	Відповідність дизайну інтерфейсу вимогам	так
16	Зручність використання для нових користувачів	так
17	Зрозумілість логіки взаємодії з користувачем	так
18	Легкодоступність всіх функцій додатку	так
19	Швидкодія додатку без затримок	так
20	Прийнятний час обробки та класифікації зображень	так

Продовження таблиці 5.1

21	Працездатність без підключення до інтернету	так
22	Доступність всіх функцій в автономному режимі	так
23	Сумісність з різними моделями пристроїв	так
24	Підтримка різних версій Android	так
25	Робота з різними форматами зображень (JPEG, PNG, BMP)	так
26	Коректність обробки зображень при різних умовах освітлення	так
27	Працездатність при низькому рівні заряду батареї	так
28	Працездатність при обмеженій пам'яті на пристрої	так
29	Коректне відображення інтерфейсу та даних при зміні орієнтації екрану	так
30	Відновлення даних після аварійного завершення роботи	так
31	Коректна робота з різними мовами інтерфейсу	так
32	Правильне відображення текстових елементів на обраній мові	так

В таблицях Д.1–Д.10 додатку Д викладено приклади розроблених тест-кейсів.

Тестування зручності використання (Usability Testing) та інтерфейсу користувача (UI Testing) показали, що мобільний додаток для класифікації зображень відповідає вимогам якості та зручності. Коректне розташування елементів інтерфейсу, логічна навігація та відповідність дизайну забезпечують комфортне використання додатку. Інтуїтивна зрозумілість та висока задоволеність користувачів підтверджують успішність реалізації функціоналу та дизайну додатку, що сприяє позитивному досвіду користувачів та підвищує їхню лояльність.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було розроблено мобільний додаток для класифікації зображень, що дозволяє користувачам, використовуючи обмежені ресурси та “поганий” Інтернет, класифікувати зображення за темами: люди, тварини, природа.

Було проведено дослідження предметної області та аналіз існуючих рішень на ринку, що показав – жодна з наявних систем не відповідає повністю потребам користувачів, зокрема щодо офлайн-функціональності та зручності використання. Тому було створено мобільний додаток на мові програмування Java, який вирішить ці проблеми та надасть користувачам можливість легко та швидко класифікувати зображення .

Додаток інтегровано у мобільний пристрій та можливо застосовувати одночасно при створенні фото та/або застосовувати обравши фото з галереї. Класифікація зображень буде виконана на навченій та після відображення результату класифікації збережено у сховище пристрою.

Додаток був протестований, є ефективним, хоч на даний момент з обмеженим переліком тем для класифікації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Cam Find app [Електронний ресурс] URL: https://camfindapp.com/ (дата звернення 10.05.2024).

2. Google Lens app [Електронний ресурс] URL: https://lens.google/ (дата звернення 09.05.2024).

3. Google Lens mobile app [Електронний ресурс] URL: https://play.google.com/store/apps/details?id=com.google.ar.lens (дата звернення 09.05.2024).

4. Google's ANN service. – Vertex AI: [Електронний ресурс] URL: https://cloud.google.com/vertex-ai/docs/matching-engine/ann-service-overview (дата звернення: 08.05.2022).

5. K. Smelyakov, Y. Honchar, O. Bohomolov and A. Chupryna., Machine Learning Models Efficiency Analysis for Image Classification Problem . // Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Systems (COLINS), Volume I: Main Conference, 2022. In CEUR Workshop Proceedings, Vol-3171, 2022, pp. 942-959.

6. Seeing AI App Launches on Android – Including new and updated features and new languages [Електронний ресурс] URL: https://blogs.microsoft.com/accessibility/seeing-ai-app-launches-on-android-including-new-and-updated-features-and-new-languages/ (дата звернення 10.05.2024).

7. Sharma S. TensorFlow on Mobile [Електронний ресурс] / Sagar Sharma. – 2018. – Режим доступу до ресурсу: https://towardsdatascience.com/tensorflow-on-mobile-tutorial-1-744703297267 (дата звернення: 27.06.2022).

8. Документація draw.io для UML проектування [Електронний ресурс] URL: https://www.drawio.com/doc/getting-started-editor (дата звернення: 05.06.2024)

9. Документація Figma [Електронний ресурс] URL: https://help.figma.com/hc/en-us/categories/360002042553 (дата звернення: 12.06.2022).

10. Документація Gradle [Електронний ресурс] URL: https://docs.gradle.org/current/userguide/userguide.html

11. Документація з tensorflow. [Електронний ресурс] URL: https://www.tensorflow.org/lite/api_docs (дата звернення: 12.05.2024).

12. Документація з тестування в Agile [Електронний ресурс] URL: https://www.agilealliance.org/

13. ДСТУ 3008:2015 Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. К.: ДП «УкрНДНЦ», 2016. 26 с. (дата звернення: 06.05.2024).

14. Кириченко І.В., Рошка В.Д. Підходи розробки ігрового штучного інтелекту // Proceedings of the 6 th International Scientific and Practical Conference SCIENTIFIC COMMUNITY: INTERDISCIPLINARY RESEARCH HAMBURG, GERMANY 26-28.01.2022 p. 1065-1069

15. Нейронні мережі – шлях до глибинного навчання – Codeguida: [Електронний ресурс] URL: https://codeguida.com/post/739 (дата звернення: 20.05.2022).

16. Основи верстки мобільних додатків. [Електронний ресурс] URL: https://habr.com/ru/articles/181820/ (дата звернення: 10.05.2024)

17. Порівняння фреймворків для глибокого навчання. [Електронний ресурс] URL: https://habr.com/ru/companies/otus/articles/443874/ (дата звернення: 22.05.2024)

18. Продукти Atlassian [Електронний ресурс] URL: https://support.atlassian.com/jira-software-cloud/resources/

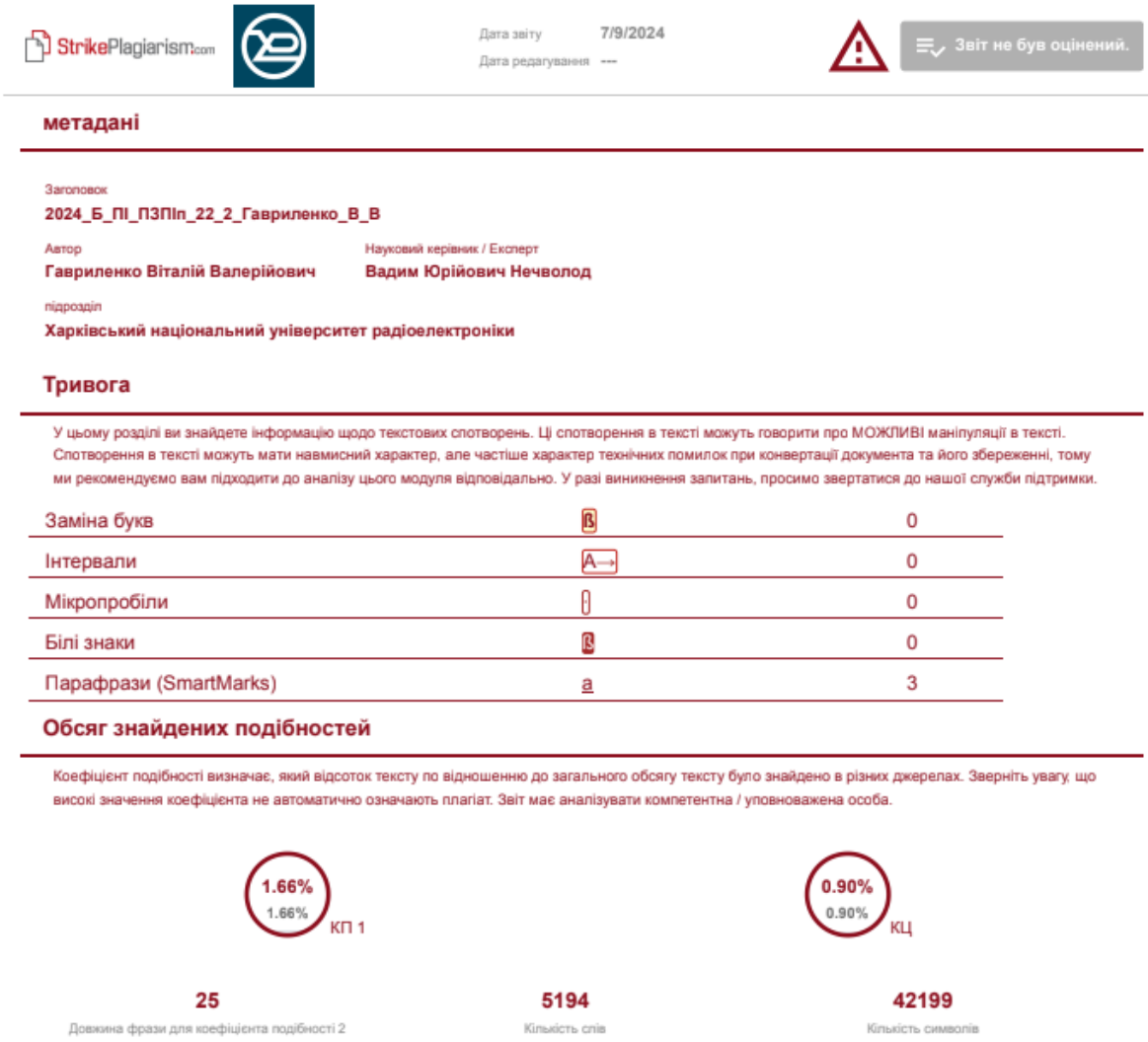
19. Смеляков К.С., Чуприна А.С., Сандркін Д.Л., Вакулік Є.В., Дроб Є.М., Розробка інваріантної моделі цифрового зображення для швидкого пошуку у сховищах даних // Збірник наукових праць ХНУПС. – № 2 (68). – 2021. – С. 108-115.

20. Специфікація стандартів по тестуванню [Електронний ресурс] URL: https://standards.ieee.org/ieee/29119-1/5308/

21. Український переклад документації Python https://github.com/python/python-docs-uk

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



StrikePlagiarism.com

Дата звіту 7/9/2024
Дата редагування ---

Звіт не був оцінений.

метадані

Заголовок
2024_Б_ПІ_ПЗПІп_22_2_Гавриленко_В_В

Автор
Гавриленко Віталій Валерійович

Науковий керівник / Експерт
Вадим Юрійович Нечволод

підрозділ
Харківський національний університет радіоелектроніки

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли	(	0
Білі знаки	ß	0
Парафрази (SmartMarks)	a	3

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

1.66% 1.66% КП 1

0.90% 0.90% КЦ

25
Довжина фрази для коефіцієнта подібності 2

5194
Кількість слів

42199
Кількість символів

ДОДАТОК Б

Перелік джерел посилання за науковими напрямками керівника та науковців кафедри програмної інженерії

5. K. Smelyakov, Y. Honchar, O. Bohomolov and A. Chupryna., Machine Learning Models Efficiency Analysis for Image Classification Problem // Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Systems (COLINS), Volume I: Main Conference, 2022. In CEUR Workshop Proceedings, Vol-3171, 2022, pp. 942–959.

14. Кириченко І.В., Рошка В.Д. Підходи розробки ігрового штучного інтелекту // Proceedings of the 6 th International Scientific and Practical Conference SCIENTIFIC COMMUNITY: INTERDISCIPLINARY RESEARCH HAMBURG, GERMANY 26-28.01.2022 p. 1065–1069.

19. Смеляков К. С., Чуприна А.С., Сандркін Д.Л., Вакулік Є.В., Дроб Є.М., Розробка інваріантної моделі цифрового зображення для швидкого пошуку у сховищах даних // Збірник наукових праць ХНУПС. – № 2 (68). – 2021. – С. 108–115.

ДОДАТОК В

Приклади інтерфейсу користувача

Рисунок В.1 – Логотип додатку (виконано самостійно)

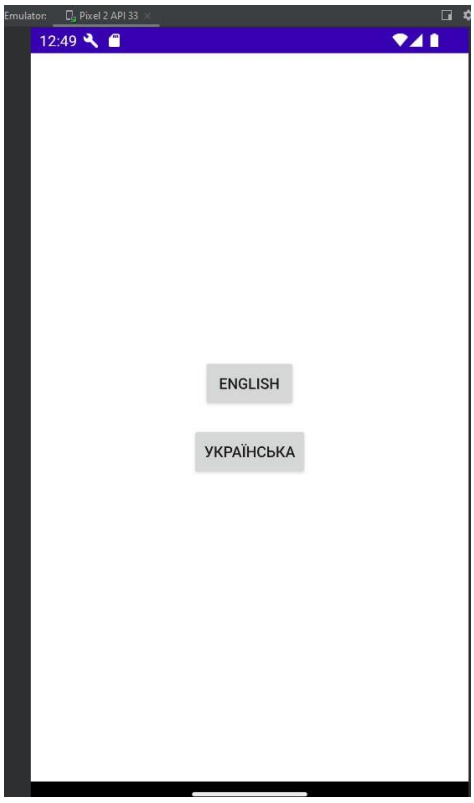


Рисунок В.2 - Сторінка обрання мови додатку (виконано самостійно)

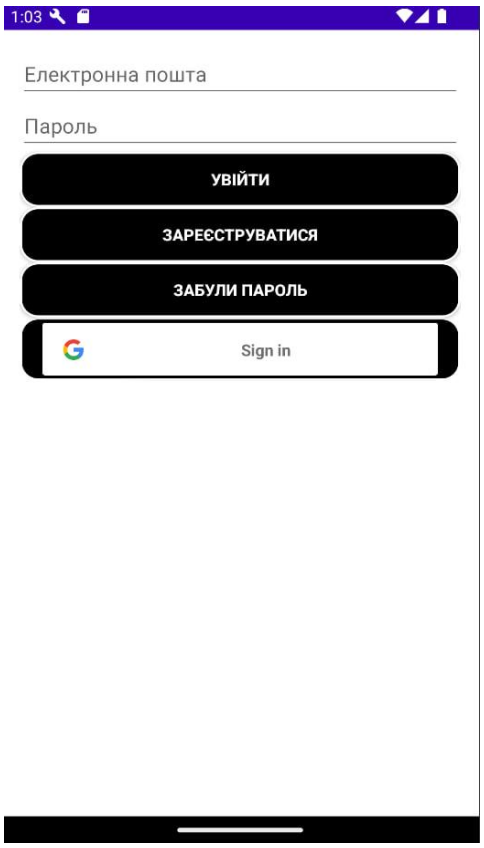


Рисунок В.3 - Сторінка вводу авторизаційних даних (виконано самостійно)

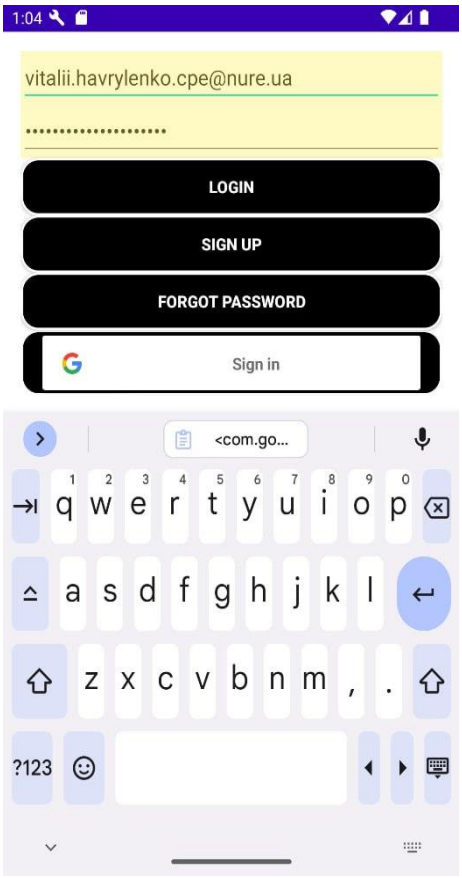


Рисунок В.4 - Приклад вводу авторизаційних даних (виконано самостійно)

Рисунок В.5 - Внутрішній інтерфейс додатку (виконано самостійно)

Рисунок В.6 - Сторінка вибору джерела зображення (виконано самостійно)

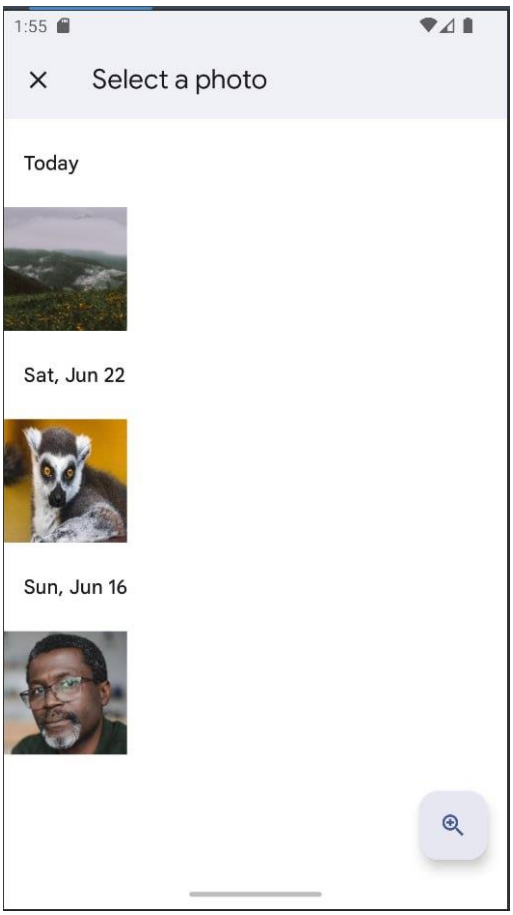


Рисунок В.7 - Екран вибору зображення для класифікації (виконано самостійно)

Рисунок В.8 - Приклад результату класифікації (виконано самостійно)

Рисунок В.9 - Приклад результату класифікації (виконано самостійно)

ДОДАТОК Г

Приклад коду програми

JSON налаштування сервісу авторизації google-services.json.


```
{
"type": "service_account",
"project_id": "imageclassifier-e108e",
"private_key_id": "9e5c7a9fb8fb515352d94acda0287da7186a3333",
"private_key":"-----BEGIN PRIVATE KEY-----
\nMIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQCRq+T9+QzTYMGm\noThh/AhG
lyGrf0BS+E6JB7gSgwxLgKn8eyH7kChLjwMrFgj+9PB8gZrf44D5kZqH\nqrLLhG3FUut1aj9fnC
6dSEF/Gnz0HKvgrMnXGdhA67xjwX21VWNiUmFv3VS8ONeM\nAEvdK1QJBuBRfzyO+wCa5l6dtOlq
AxGG+PM1S/vbVK4kX249ZQcGWEgBWNJc4FDs\n8BCDoOmvnsqjY+YQ7HolMUb5eq+rosFemViCM4
gNVXn0CilylYCHHTpX1nQTrqpU\ncXbkpyPrNisHonoolAA4JnNrGBPr8MOiHPdpwwXv41hHPQC/
hySXhO+NX+Kj5YmL\nRZoud+ubAgMBAAECggEAEdymA45aSEEA/dGe/ETPLvixJDpCROo11w/fPk
XbaBaq\nwU7H5/TRPBKZLNtS1xBpvJqXxwQ6eF+mNyRzMAlY/G0ieg9BHhQkkxydC1zQHY62\ndK
I7TajZn3xIaWL8axPBT/kSAYl+LmY55+zRYJlDgKfbMaMMZevZMyU0TNo2FM2E\nJlEODlJt9JDF
NJmz85ClvVVI8SPyfwNVEOUAatWBSqRgGV66CDzLnG6iLH4sK5PZ\nTTRNxmNkBRVUh1t/s5Q9Gq
D1+8i2AaJMgJarQ+h9KzXdXIM9kpI9TsM83PP1siL5\ntktg+KBUtP0GQHh5Lm8a+mfHWoOZNbO/
qHeBzUxqQQKBgQDFJ2DQDhTFbmAROGVP\nmUspm7N31AeOYmc0b7HKbRkrH+KRwFTgDn817L9QjX
uGaO/aT6EmfhIZu4orhAx4\n098O/IhGVcqjydEsIXdubwaCDaf27dm30pg/OpAuM/yX1jUJKxxO
AzWUQG5hifDj\nmSsRA5oFnmzB/43UETL7Yt1CiQKBgQC9JrlBoMPh+Urzp9atciF4DKJR0eycMN
N0\nR2dDatNN6WP0qHrYbHurpto9754KfrdVoZYcA1QyipVahWXT0a9pzey7Bkr1zf8x\nZXbJz5
k1DX+onsxjWGkW+Lo3cnZBZuH2D1651qV8OEIEdDHwVJs8LcLYh8R/iIh4\nTiMdDKYEAwKBgENQ
+7xuYmc7FbArimQQWlqdoDy099OFA+oijRkqASL5YxoTCqfq\nK0NVT8mIVXoCUItBDywedb21jC
rpjnyfExWB4mHpnyQnn7UqxCW/4P/8+7HTarha\nPsOwElTESWXqrPcxchus+tlg2o+RUt/Th6Hq
keeDPW1L/bhAGyE8k4HJAoGAFcpm\nQTdCGcBccCF/y8+AJ5JdCMwEVCIT2it3q/6chTdD+qWGMX
Sd8rJwJf562H6spELP\nRj7j7kak2h7QEkzvxmKgyRL6g8LNiWVFAA7YeZIqiMRyi2deA/V3jnez
Q+mUUoKG\nySX/PoZEpLq2HhfBInerIMLTUlXqUuG9jdUtMv0CgYBBt0bUc2CxJO1Cx7Fche/5\n
xyUch1ZuoG5VUKHAdB18oR/wUu0wnjzNPys7f3zyDdk7Tsdq+V+gHmEz6G/eMp8F\nCUqQ/fp8QV
iT2TkQsySa6v9VeAz1KhybTcCz0/zv+ME27IM+vifQAGUFDZRVeCKf\nfF3cz2Fm0dU5T4thmSO7
Cg==\n-----END PRIVATE KEY-----\n",
"client_email": "firebase-adminsdk-ov47s@imageclassifier-
e108e.iam.gserviceaccount.com",
"client_id": "114998127429774437678",
"auth_uri": "https://accounts.google.com/o/oauth2/auth",
"token_uri": "https://oauth2.googleapis.com/token",
"auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
"client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-
ov47s%40imageclassifier-e108e.iam.gserviceaccount.com",
"universe_domain": "googleapis.com"
}
```


Код основного класу додатку.

```
public class MainActivity extends AppCompatActivity {
    private static final int CAMERA_PERMISSION_REQUEST_CODE = 200;
    private ImageView imageView;
    private TextView tvWelcome, tvInstruction, classificationResult;
    private Button btnCamera, btnGallery, btnSave, btnCancel;
```

```
    private FloatingActionButton fabClose;
    private Bitmap currentImage;
    private String currentImagePath;
    private FirebaseAuthManager firebaseAuthManager;
    private ImageProcessor imageProcessor;
    private final ActivityResultLauncher<Intent> cameraLauncher =
registerForActivityResult(
            new ActivityResultContracts.StartActivityForResult(),
            result -> {
                if (result.getResultCode() == RESULT_OK) {
                    Intent data = result.getData();
                    if (data != null) {
                        currentImage = (Bitmap)
data.getExtras().get("data");
                        currentImagePath =
imageProcessor.saveTempImage(currentImage);
                        displayImageAndClassify();
                    }
                }
            }
    );
    private final ActivityResultLauncher<Intent> galleryLauncher =
registerForActivityResult(
            new ActivityResultContracts.StartActivityForResult(),
            result -> {
                if (result.getResultCode() == RESULT_OK) {
                    Intent data = result.getData();
                    if (data != null) {
                        Uri imageUri = data.getData();
                        currentImagePath = imageUri.toString();
                        loadImageFromUri(imageUri);
                    }
                }
            }
    );
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_settings, menu);
        return true;
    }
    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if (item.getItemId() == R.id.action_change_language) {
            startActivity(new Intent(this, LanguageSettingsActivity.class));
            return true;
        }
        return super.onOptionsItemSelected(item);
    }
```

```
    @RequiresApi(api = Build.VERSION_CODES.N)
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        setTheme(R.style.AppTheme);
        SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(this);
        String language = prefs.getString("app_language", null);
        if (language == null) {
            Intent intent = new Intent(this,
LanguageSettingsActivity.class);
            startActivity(intent);
            finish();
            return;
        }
        setLocale(language);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        firebaseAuthManager = new FirebaseAuthManager(this);
        if (!firebaseAuthManager.isUserLoggedIn()) {
            firebaseAuthManager.redirectToLogin();
            finish();
            return;
        }
        imageProcessor = new ImageProcessor(this);
        imageView = findViewById(R.id.imageView);
        tvWelcome = findViewById(R.id.tvWelcome);
        tvInstruction = findViewById(R.id.tvInstruction);
        classificationResult = findViewById(R.id.classificationResult);
        btnCamera = findViewById(R.id.btnCamera);
        btnGallery = findViewById(R.id.btnGallery);
        btnSave = findViewById(R.id.btnSave);
        btnCancel = findViewById(R.id.btnCancel);
        fabClose = findViewById(R.id.fabClose);
        btnCamera.setVisibility(View.GONE);
        btnGallery.setVisibility(View.GONE);
        imageView.setVisibility(View.GONE);
        tvInstruction.setVisibility(View.GONE);
        classificationResult.setVisibility(View.GONE);
        btnSave.setVisibility(View.GONE);
        btnCancel.setVisibility(View.GONE);

new Handler(Looper.getMainLooper()).postDelayed(() -> {
            tvWelcome.setVisibility(View.GONE);
            tvInstruction.setVisibility(View.VISIBLE);
            btnCamera.setVisibility(View.VISIBLE);
            btnGallery.setVisibility(View.VISIBLE);
        }, 5000);
```

```
        btnCamera.setOnClickListener(v -> checkCameraPermission());
        btnGallery.setOnClickListener(v -> openGallery());
        fabClose.setOnClickListener(v -> finish());
        btnSave.setOnClickListener(v -> saveImage());
        btnCancel.setOnClickListener(v -> cancelClassification());
    }
    /
     * Перевіряє дозвіл на використання камери.
     * Якщо дозволу немає, запитує його.
     */
    private void checkCameraPermission() {
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.CAMERA}, CAMERA_PERMISSION_REQUEST_CODE);
        } else {
            openCamera();
        }
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if (requestCode == CAMERA_PERMISSION_REQUEST_CODE) {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                openCamera();
            } else {
                Toast.makeText(this, "Дозвіл на використання камери
відхилено", Toast.LENGTH_LONG).show();
            }
        }
    }
    /
     * Відкриває камеру для зйомки зображення.
     */
    private void openCamera() {
        Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        cameraLauncher.launch(cameraIntent);
    }
    /
     * Відкриває галерею для вибору зображення.
     */
    private void openGallery() {
        Intent galleryIntent = new Intent(Intent.ACTION_PICK,
MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
        galleryLauncher.launch(galleryIntent);
    }
```

```
    /
     * Завантажує зображення з Uri і відображає його.
     *
     * @param uri URI зображення.
     */
    private void loadImageFromUri(Uri uri) {
        try {
            InputStream imageStream =
getContentResolver().openInputStream(uri);
            currentImage = BitmapFactory.decodeStream(imageStream);
            if (currentImage != null) {
                displayImageAndClassify();
            } else {
                Toast.makeText(this,
getString(R.string.failed_to_load_image), Toast.LENGTH_LONG).show();
            }
        } catch (IOException e) {
            Toast.makeText(this, "Помилка при завантаженні зображення",
Toast.LENGTH_LONG).show();
            e.printStackTrace();
        }
    }
/
     * Відображає зображення та виконує його класифікацію.
     */
    private void displayImageAndClassify() {
        if (currentImage != null) {
            imageView.setImageBitmap(currentImage);
            imageView.setVisibility(View.VISIBLE);
            tvInstruction.setVisibility(View.GONE);
            btnCamera.setVisibility(View.GONE);
            btnGallery.setVisibility(View.GONE);
            classifyImageAsync(currentImage);
            classificationResult.setVisibility(View.VISIBLE);
            btnSave.setVisibility(View.VISIBLE);
            btnCancel.setVisibility(View.VISIBLE);
        } else {
            Toast.makeText(this, "Зображення не вдалося завантажити",
Toast.LENGTH_LONG).show();
        }
    }
    /
     * Виконує класифікацію зображення асинхронно.
     *
     * @param bitmap зображення для класифікації.
     */
    private void classifyImageAsync(Bitmap bitmap) {
        ExecutorService executor = Executors.newSingleThreadExecutor();
        Handler handler = new Handler(Looper.getMainLooper());
```

```
        executor.execute(() -> {
            Bitmap resizedBitmap = Bitmap.createScaledBitmap(bitmap, 224,
224, false);
            String result = imageProcessor.classifyImage(resizedBitmap);
            handler.post(() -> {
                classificationResult.setText(result);
            });
        });
    }
...
```

Код класу моделі зображення.

```
public class Model {
    private final Interpreter interpreter;
    private Model(Interpreter interpreter) {
        this.interpreter = interpreter;
    }
    public static Model newInstance(Context context) throws IOException {
        MappedByteBuffer modelBuffer = loadModelFile(context);
        Interpreter interpreter = new Interpreter(modelBuffer);
        return new Model(interpreter);
    }
    public Outputs process(TensorBuffer inputBuffer) {
        TensorBuffer outputBuffer = TensorBuffer.createFixedSize(new int[]{1,
3}, DataType.FLOAT32);
        interpreter.run(inputBuffer.getBuffer(), outputBuffer.getBuffer());
        return new Outputs(outputBuffer);
    }
    public void close() {
        interpreter.close();
    }
    private static MappedByteBuffer loadModelFile(Context context) throws
IOException {
        InputStream inputStream =
context.getAssets().open("image_classifier_model.tflite");
        File tempFile = File.createTempFile("model", ".tflite",
context.getCacheDir());
        try (FileOutputStream out = new FileOutputStream(tempFile)) {
            byte[] buffer = new byte[1024];
            int bytesRead;
            while ((bytesRead = inputStream.read(buffer)) != -1) {
                out.write(buffer, 0, bytesRead);
            }
        }
        try (FileInputStream fis = new FileInputStream(tempFile);
             FileChannel fileChannel = fis.getChannel()) {
            return fileChannel.map(FileChannel.MapMode.READ_ONLY, 0,
fileChannel.size());
```

```
        }
    }
    public static class Outputs {
        private final TensorBuffer outputBuffer;
        public Outputs(TensorBuffer outputBuffer) {
            this.outputBuffer = outputBuffer;
        }
        public TensorBuffer getOutputFeature0AsTensorBuffer() {
            return outputBuffer;
        }
    }
}
```

Код класу аутентифікації.

```
public class FirebaseAuthManager {
    private FirebaseAuth mAuth;
    private Context context;
    /
     * Конструктор, який ініціалізує FirebaseAuth та контекст.
     *
     * @param context контекст, в якому працює менеджер аутентифікації.
     */
    public FirebaseAuthManager(Context context) {
        this.context = context;
        this.mAuth = FirebaseAuth.getInstance();
    }
    /
     * Перевіряє, чи користувач вже увійшов в систему.
     *
     * @return true, якщо користувач увійшов в систему, інакше false.
     */
    public boolean isUserLoggedIn() {
        return mAuth.getCurrentUser() != null;
    }
    /
     * Перенаправляє користувача на екран входу.
     */
    public void redirectToLogin() {
        Intent intent = new Intent(context, LoginActivity.class);
        context.startActivity(intent);
    }
}
```

Код класу локалізації.

```
public class LanguageSettingsActivity extends AppCompatActivity {
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_language_settings);
        Button btnEnglish = findViewById(R.id.btnEnglish);
        Button btnUkrainian = findViewById(R.id.btnUkrainian);
        btnEnglish.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                setLocale("en");
            }
        });
        btnUkrainian.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                setLocale("uk");
            }
        });
    }
    private void setLocale(String lang) {
        Locale locale = new Locale(lang);
        Locale.setDefault(locale);
        Configuration config = new Configuration();
        config.setLocale(locale);
        getResources().updateConfiguration(config,
getResources().getDisplayMetrics());
        SharedPreferences.Editor                    editor                    =
PreferenceManager.getDefaultSharedPreferences(this).edit();
        editor.putString("app_language", lang);
        editor.apply();
        // Перезапуск активності, щоб застосувати зміну мови
        Intent intent = new Intent(this, LoginActivity.class);
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        startActivity(intent);
        finish();
    }
}
```

Код класу авторизації.

```
public class LoginActivity extends AppCompatActivity {
    private static final int RC_SIGN_IN = 9001;
    private FirebaseAuth mAuth;
    private EditText emailEditText;
    private EditText passwordEditText;
    private Button loginButton;
    private Button signupButton;
    private Button resetPasswordButton;
    private GoogleSignInClient mGoogleSignInClient;
    @Override
```

```
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        setTheme(R.style.AppTheme);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        // Ініціалізація Firebase
        FirebaseApp.initializeApp(this);
        mAuth = FirebaseAuth.getInstance();
        emailEditText = findViewById(R.id.emailEditText);
        passwordEditText = findViewById(R.id.passwordEditText);
        loginButton = findViewById(R.id.loginButton);
        signupButton = findViewById(R.id.signupButton);
        resetPasswordButton = findViewById(R.id.resetPasswordButton);
        loginButton.setOnClickListener(v -> {
            String email = emailEditText.getText().toString().trim();
            String password = passwordEditText.getText().toString().trim();
            if (validateInputs(email, password)) {
                signIn(email, password);
            }
        });
        signupButton.setOnClickListener(v -> {
            String email = emailEditText.getText().toString().trim();
            String password = passwordEditText.getText().toString().trim();
            if (validateInputs(email, password)) {
                signUp(email, password);
            }
        });
        resetPasswordButton.setOnClickListener(v -> {
            String email = emailEditText.getText().toString().trim();
            if (!email.isEmpty()) {
                resetPassword(email);
            } else {
                Toast.makeText(this, "Будь ласка, введіть електронну пошту для
відновлення пароля", Toast.LENGTH_SHORT).show();
            }
        });
        // Налаштування Google Sign-In
        GoogleSignInOptions              gso              =              new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
                .requestIdToken(getString(R.string.default_web_client_id))
                .requestEmail()
                .build();
        mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
        findViewById(R.id.googleSignInButton).setOnClickListener(v         ->
signInWithGoogle());
    }
    private boolean validateInputs(String email, String password) {
        if (email.isEmpty()) {
            Toast.makeText(this,  "Будь  ласка,  введіть  електронну  пошту",
Toast.LENGTH_SHORT).show();
```

```
            return false;
        }
        if (password.isEmpty()) {
            Toast.makeText(this,     "Будь     ласка,     введіть     пароль",
Toast.LENGTH_SHORT).show();
            return false;
        }
        return true;
    }
    private void signIn(String email, String password) {
        mAuth.signInWithEmailAndPassword(email, password)
                .addOnCompleteListener(this, task -> {
                    if (task.isSuccessful()) {
                        FirebaseUser user = mAuth.getCurrentUser();
                        Log.d("LoginActivity", "signInWithEmail:success " +
user.getEmail());
                        Toast.makeText(LoginActivity.this,    "Authentication
Successful.", Toast.LENGTH_SHORT).show();
                        navigateToMainActivity();
                    } else {
                        Log.w("LoginActivity",     "signInWithEmail:failure",
task.getException());
                        Toast.makeText(LoginActivity.this,    "Authentication
Failed.", Toast.LENGTH_SHORT).show();
                    }
                });
    }
```

Розмітка сторінки логіна.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="16dp"
    android:background="@android:color/white"
    tools:context=".LoginActivity">
    <EditText
        android:id="@+id/emailEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/email" />
    <EditText
        android:id="@+id/passwordEditText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="@string/password"
```

```
        android:inputType="textPassword" />
    <Button
        android:id="@+id/loginButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/login"
        style="@style/RoundedButtonStyle" />
    <Button
        android:id="@+id/signupButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/sign_up"
        style="@style/RoundedButtonStyle" />
    <Button
        android:id="@+id/resetPasswordButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/reset_password"
        style="@style/RoundedButtonStyle" />
    <com.google.android.gms.common.SignInButton
        android:id="@+id/googleSignInButton"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/google_sign_in"
        style="@style/RoundedButtonStyle" />
</LinearLayout>
```

Приклад коду локалізації.

```
    <string name="app_name"></string>
    <string name="close_app">Закрити додаток</string>
  <string name="minimum_password">Пароль занадто короткий, введіть щонайменше
6 символів!</string>
    <string name="auth_failed">Авторизація не вдалася</string>
    <string name="email">Електронна пошта</string>
    <string name="password">Пароль</string>
    <string name="login">Увійти</string>
    <string name="sign_up">Зареєструватися</string>
    <string name="back">Назад</string>
    <string name="enter_email">Введіть електронну пошту!</string>
    <string name="enter_password">Введіть пароль!</string>
    <string name="reset_password">Забули пароль</string>
    <string name="google_sign_in">Увійти через Google</string>
    <string   name="toast_enter_email">Будь   ласка,   введіть   електронну
пошту</string>
    <string name="toast_enter_password">Будь ласка, введіть пароль</string>
    <string name="toast_auth_success">Авторизація успішна</string>
    <string name="toast_auth_failed">Авторизація не вдалася</string>
    <string name="toast_registration_success">Реєстрація успішна</string>
```

```
    <string name="toast_registration_failed">Реєстрація не вдалася</string>
    <string name="change_language">Змінити мову</string>
    <string name="welcome_message">Ласкаво просимо до ImageClassifier!</string>
    <string name="instruction_message">Обери секцію для продовження роботи. \n Додай кольорів у своє життя!!!</string>
    <string name="camera_permission_denied">Дозвіл на використання камери відхилено</string>
    <string name="failed_to_load_image">Не вдалося завантажити зображення</string>
    <string name="error_loading_image">Помилка при завантаженні зображення</string>
    <string name="classification_result">Результат класифікації: </string>
    <string name="error_processing_image">Помилка при обробці зображення</string>
    <string name="image_saved">Зображення збережено в </string>
    <string name="error_saving_image">Помилка при збереженні зображення</string>
    <string name="take_photo">Зробити фото</string>
    <string name="select_photo_from_gallery">Вибрати фото з галереї</string>
    <string name="save">Зберегти</string>
    <string name="cancel">Відмінити</string>
    <string name="toast_reset_password_email_sent">Інструкції для відновлення пароля надіслано на електронну пошту</string>
    <string name="toast_reset_password_failed">Помилка при відновленні пароля</string>
    <string name="sign_in_with_google">Увійти через Google</string>
</resources>
```

ДОДАТОК Д

Приклади розроблених тест-кейси

Таблиця Д.1 – Тест-кейс “Перевірка реєстрації нового користувача” (виконана самостійно)

ID кейсу	1
Назва кейсу	Перевірка реєстрації нового користувача
Опис	Перевірка можливості реєстрації нового користувача з валідними даними
Передумови	Додаток запущений, користувач знаходиться на екрані реєстрації
Кроки	1. Ввести валідну адресу електронної пошти 2. Ввести валідний пароль 3. Підтвердити пароль 4. Натиснути кнопку 'Зареєструватися'
Очікуваний результат	Користувач успішно зареєстрований, перенаправлений на головний екран

Таблиця Д.2 – Тест-кейс “Перевірка реєстрації зі вже існуючою електронною поштою” (виконана самостійно)

ID кейсу	2
Назва кейсу	Перевірка реєстрації зі вже існуючою електронною поштою
Опис	Перевірка відмови у реєстрації з уже існуючою електронною поштою
Передумови	Додаток запущений, користувач знаходиться на екрані реєстрації
Кроки	1. Ввести адресу електронної пошти, яка вже зареєстрована 2. Ввести валідний пароль 3. Підтвердити пароль 4. Натиснути кнопку 'Зареєструватися'
Очікуваний результат	Відображається повідомлення про помилку, що адреса електронної пошти вже зареєстрована

Таблиця Д.3 – Тест-кейс “Перевірка авторизації з валідними даними” (виконана самостійно)

ID кейсу	3
Назва кейсу	Перевірка авторизації з валідними даними
Опис	Перевірка можливості авторизації з валідною електронною поштою та паролем
Передумови	Додаток запущений, користувач знаходиться на екрані авторизації
Кроки	1. Ввести валідну адресу електронної пошти 2. Ввести валідний пароль 3. Натиснути кнопку 'Увійти'
Очікуваний результат	Користувач успішно авторизується і потрапляє на головний екран

Таблиця Д.4 – Тест-кейс “Перевірка авторизації з невірним паролем” (виконана самостійно)

ID кейсу	4
Назва кейсу	Перевірка авторизації з невірним паролем
Опис	Перевірка відмови у авторизації з невірним паролем
Передумови	Додаток запущений, користувач знаходиться на екрані авторизації
Кроки	1. Ввести валідну адресу електронної пошти 2. Ввести невірний пароль 3. Натиснути кнопку 'Увійти'
Очікуваний результат	Відображається повідомлення про невірний пароль

Таблиця Д.5 – Тест-кейс “Перевірка авторизації з невірною електронною поштою” (виконана самостійно)

ID кейсу	5
Назва кейсу	Перевірка авторизації з невірною електронною поштою

Продовження таблиці Д.5

Опис	Перевірка відмови у авторизації з невірною електронною поштою
Передумови	Додаток запущений, користувач знаходиться на екрані авторизації
Кроки	1. Ввести невірну адресу електронної пошти 2. Ввести валідний пароль 3. Натиснути кнопку 'Увійти'
Очікуваний результат	Відображається повідомлення про невірну адресу електронної пошти

Таблиця Д.6 – Тест-кейс “Перевірка можливості виходу з облікового запису” (виконана самостійно)

ID кейсу	6
Назва кейсу	Перевірка можливості виходу з облікового запису
Опис	Перевірка можливості виходу з облікового запису після авторизації
Передумови	Користувач авторизований і знаходиться на головному екрані
Кроки	1. Натиснути на іконку профілю 2. Натиснути кнопку 'Вийти'
Очікуваний результат	Користувач успішно виходить з облікового запису і повертається на екран авторизації

Таблиця Д.7 – Тест-кейс “Перевірка відновлення паролю” (виконана самостійно)

ID кейсу	7
Назва кейсу	Перевірка відновлення паролю
Опис	Перевірка функціоналу відновлення паролю через електронну пошту
Передумови	Додаток запущений, користувач знаходиться на екрані відновлення паролю

Продовження таблиці Д.7

Кроки	1. Ввести зареєстровану адресу електронної пошти 2. Натиснути кнопку 'Відновити пароль' 3. Перевірити електронну пошту на наявність листа з інструкціями
Очікуваний результат	Користувач отримує лист з інструкціями для відновлення паролю

Таблиця Д.8 – Тест-кейс “Перевірка доступу до головного екрану без авторизації” (виконана самостійно)

ID кейсу	8
Назва кейсу	Перевірка доступу до головного екрану без авторизації
Опис	Перевірка, що користувач не може отримати доступ до головного екрану без авторизації
Передумови	Додаток запущений, користувач не авторизований
Кроки	1. Спробувати отримати доступ до головного екрану додатку без авторизації
Очікуваний результат	Користувач не може отримати доступ до головного екрану без авторизації

Таблиця Д.9 – Тест-кейс “Перевірка доступу до функцій класифікації без авторизації” (виконана самостійно)

ID кейсу	9
Назва кейсу	Перевірка доступу до функцій класифікації без авторизації
Опис	Перевірка, що користувач не може використовувати функції класифікації без авторизації
Передумови	Додаток запущений, користувач не авторизований
Кроки	1. Спробувати скористатися функцією класифікації зображень без авторизації
Очікуваний результат	Користувач не може використовувати функції класифікації без авторизації

Таблиця Д.10 – Тест-кейс “Перевірка роботи додатку після авторизації” (виконана самостійно)

ID кейсу	10
Назва кейсу	Перевірка роботи додатку після авторизації
Опис	Перевірка коректної роботи всіх функцій додатку після успішної авторизації
Передумови	Користувач авторизований і знаходиться на головному екрані
Кроки	1. Виконати основні функції додатку: зйомка, завантаження, класифікація зображень
Очікуваний результат	Всі функції додатку працюють коректно після авторизації

ДОДАТОК Е

Специфікація програмного продукту

1 ВСТУП

1.1 Огляд проєкту

Зображення відіграють важливу роль у нашому повсякденному житті, особливо в епоху цифрових технологій. Сучасні мобільні телефони з високоякісними камерами дозволяють робити та зберігати тисячі зображень. Проте організація та керування такою кількістю зображень стає дедалі складнішим завданням для користувачів. Проблема полягає в тому, що зберігання великої кількості зображень займає значний обсяг пам’яті, а пошук потрібного зображення може бути трудомістким і неефективним.

Мобільний додаток для класифікації зображень на основі штучного інтелекту спрямований на розв’язання цих проблем шляхом автоматичної класифікації та організації зображень. Додаток використовує машинне навчання для розпізнавання та категоризації зображень, що дозволяє користувачам швидко знаходити потрібні фотографії та ефективно керувати своїми галереями. Завдяки використанню сучасних технологій, таких як TensorFlow Lite, додаток забезпечує високу точність класифікації та швидкість обробки зображень, що робить його незамінним інструментом для користувачів, які бажають упорядкувати свої цифрові колекції.

1.2 Мета

Основною метою даного проєкту є створення інноваційного мобільного додатку, який забезпечить користувачам можливість автоматичної класифікації зображень за допомогою штучного інтелекту. Додаток має не лише полегшити процес організації та пошуку зображень, але й покращити загальний досвід користувачів при роботі з великою кількістю фотографій. Використання машинного навчання дозволяє додатку автоматично розпізнавати та класифікувати зображення за певними категоріями (люди, тварини, природа), а також додавати відповідні теги на основі вмісту зображень.

Крім того, додаток забезпечує стиснення зображень без помітної втрати якості, що сприяє економії місця на пристрої користувача. Інтеграція з Firebase надає можливість створення та управління обліковими записами користувачів, забезпечуючи безпечне зберігання та доступ до їхніх даних. Таким чином, додаток сприяє підвищенню ефективності обробки та управління зображеннями, відповідаючи потребам як індивідуальних користувачів, так і бізнесів.

1.3 Область застосування

Застосунок доступний на будь-якому мобільному пристрої з операційною системою Android. Використання додатку можливе як для особистого зберігання зображень, так і для комерційного використання в різних сферах, таких як освіта, медицина, креативна індустрія тощо.

1.4 Короткий зміст

Додаток зосереджений на автоматичній обробці зображень, включаючи:

– класифікацію зображень за темами (люди, тварини, природа);
– додавання тегів на основі вмісту;
– розпізнавання тексту на зображеннях;
– стиснення зображень для ефективного зберігання.

Додаткові функції додатку:

– створення облікового запису користувача;
– пошук за назвою та тегами зображення;
– завантаження та зберігання багатьох зображень;
– дані про зайнятий простір;
– можливість поділитися альбомом з іншими користувачами.

Нефункціональні вимоги:

– безпека даних;
– низька затримка відповіді системи;
– зручний та інтуїтивний інтерфейс

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Розроблювана програмна система надає можливість користувачам створювати обліковий запис та завантажувати у додаток власні зображення. Додаток автоматично обробляє та аналізує зображення для стиснення, класифікації, присвоєння тегів та розпізнавання тексту. Це задовольнить потреби широкого кола користувачів.

2.2 Функції продукту

Основні функції додатку включають:

– реєстрація та авторизація користувача;
– завантаження фото;
– класифікація зображень;
– присвоєння тегів та розпізнавання тексту;
– перегляд зображень в альбомах;
– зміна особистих даних та даних про зображення;
– завантаження всіх зображень з альбому.

2.3 Характеристики користувачів

Основною цільовою аудиторією додатку є:

– користувачі, які хочуть безпечно зберігати та впорядковувати свої особисті зображення;
– аматори та професійні фотографи, яким потрібна платформа для зберігання своїх зображень;
– компанії, яким потрібен репозиторій для зберігання та управління зображеннями документації або реклами.

2.4 Загальні обмеження

Вимоги до апаратного забезпечення:

– додаток призначений для роботи на Android-пристроях із мінімальною версією ОС Android 6.0 (Marshmallow) або вище;

– пристрій повинен мати камеру з мінімальною роздільною здатністю 8 Мп для забезпечення прийнятної якості зображень для класифікації.

Обмеження по пам'яті:

– додаток може вимагати значного обсягу оперативної пам'яті для обробки зображень, особливо при використанні моделей машинного навчання;
– необхідно забезпечити достатній обсяг вільної пам'яті на пристрої для тимчасового зберігання зображень та моделей машинного навчання.

Обмеження по продуктивності:

– швидкість класифікації може варіюватися в залежності від продуктивності пристрою;
– використання моделі машинного навчання може спричинити підвищене навантаження на процесор та батарею пристрою.

2.5 Припущення й залежності

Правильність вхідних даних:

– припускається, що користувачі будуть використовувати чіткі та фокусовані зображення для класифікації;
– припускається, що користувачі будуть надавати дозвіл на доступ до камери та галереї для повної функціональності додатку.
– Інтерфейс користувача:
– припускається, що користувачі мають базові знання з використання мобільних додатків та розуміють мову інтерфейсу (англійську або українську).

Модель машинного навчання:

– припускається, що завантажена модель машинного навчання коректно навчена та оптимізована для використання на мобільних пристроях;
– припускається, що модель здатна коректно класифікувати категорії, для яких вона була навчена.

Бібліотеки та фреймворки:

– додаток використовує TensorFlow Lite для виконання класифікації зображень;
– використання Firebase для аутентифікації користувачів.

Зовнішні ресурси:

– модель машинного навчання (image_classifier_model.tflite) повинна бути доступна в ресурсах додатку;
– додаток вимагає доступу до камери та галереї для захоплення та вибору зображень.

Конфігурації та налаштування:

– додаток використовує SharedPreferences для зберігання налаштувань мови та інших параметрів користувача;
– налаштування доступу до камери та галереї повинні бути надані користувачем для коректної роботи додатку.

Мережеві залежності:

– незважаючи на те, що основна функціональність додатку не залежить від інтернет-з'єднання, доступ до Firebase для аутентифікації вимагає активного інтернет-з'єднання.

3 КОНКРЕТНІ ВИМОГИ

3.1 Вимоги до інтерфейсів

3.1.1. Інтерфейс користувача

В якості клієнтської частини буде виступати мобільний застосунок для системи Android, написаний за допомогою мови програмування Java та Android SDK. До основних функціональних вимог клієнтської частини можна віднести наступні:

– авторизація користувача: користувач може зареєструватись, а вже зареєстрований користувач авторизуватись в застосунку за допомогою авторизаційних даних Google, в разі втрати авторизаційних даних вони можуть бути відновлені за допомогою електронної пошти;
– локалізація додатку: користувач може обирати локалізацію застосунку (англійською або українською мовами);

– класифікація зображень за темами: застосунок буде мати можливість класифікувати зображення за темами (наприклад люди, тварини, природа) за допомогою камери та галереї мобільного пристрою;
– збереження зображень у відповідних теках: зображення повинні автоматично зберігатися у відповідних теках на основі результатів класифікації. Наприклад, зображення людей зберігатимуться в теці "люди", зображення тварин – в теці "тварини", а зображення природи – в теці "природа";
– інтуїтивно зрозумілий інтерфейс користувача: розробка зручного у використанні інтерфейсу, який дозволяє легко взаємодіяти із застосунком та швидко отримувати результати класифікації.

3.1.3. Комунікаційний протокол

В якості комунікаційних протоколів використано HTTP/HTTPS.

3.1.4. Вимоги до пам’яті

Мінімальні вимоги до пам’яті:
– оперативна пам’ять: 1 Гб;
– дискова пам’ять: 500 Мб

3.2 Атрибути програмного продукту

3.2.1 Безпека

Безпека є одним з ключових аспектів мобільного додатку для класифікації зображень. Для забезпечення надійного захисту даних користувачів, додаток використовує Firebase Authentication для аутентифікації та авторизації користувачів. Це дозволяє гарантувати, що доступ до облікових записів користувачів мають лише авторизовані особи. Аутентифікація здійснюється за допомогою JSON веб-токенів (JWT), що забезпечує безпечний обмін інформацією між клієнтом і сервером.

Крім того, всі дані, що передаються між клієнтською частиною додатку та сервером, шифруються за допомогою протоколів HTTPS, що захищає дані від

перехоплення під час передачі через Інтернет. Для зберігання зображень використовується Firebase Firestore, яка також забезпечує шифрування даних як у стані спокою, так і під час передачі. Реалізовані інтерфейси для рольового контролю доступу та керування дозволами, що дозволяє забезпечити різні рівні доступу до функцій додатку та даних в залежності від ролі користувача.

3.2.2 Супроводжуваність

Супроводжуваність програмного продукту є важливим аспектом для забезпечення його довгострокової ефективності та актуальності. Для цього проект супроводжується за допомогою системи контролю версій Git, що дозволяє відстежувати зміни в коді, співпрацювати з іншими розробниками та забезпечувати контроль якості коду. Відкритий репозиторій на GitHub дозволяє легко залучати нових розробників до проекту, забезпечуючи прозорість і доступність коду.

Регулярний рефакторинг коду проводиться для покращення його структури та читабельності, що сприяє зниженню технічного боргу та підвищенню якості програмного забезпечення. Постійний пошук та виправлення помилок здійснюється під час додавання нових компонентів, що забезпечує стабільну роботу додатку. Також відстежуються та оновлюються бібліотеки та залежності, що використовуються в проекті, для забезпечення сумісності та безпеки.

Документація проекту включає опис архітектури, специфікації, інструкції з встановлення та використання, а також коментарі в коді, що полегшує розуміння та підтримку коду іншими розробниками.

3.2.3 Переносимість

Переносимість програмного продукту забезпечується дотриманням стандартів методів кодування та використанням REST API для реалізації функцій додатку. Використання стандартних форматів даних, таких як JSON, і протоколів, таких як HTTP/HTTPS, забезпечує сумісність додатку з різними системами та платформами.

Код додатку написаний з урахуванням крос-платформенної сумісності, що дозволяє легко адаптувати його для різних операційних систем та пристроїв.

Використання бібліотек з відкритим вихідним кодом і популярних фреймворків, таких як TensorFlow Lite і Firebase, забезпечує можливість легкого перенесення додатку на інші платформи або інтеграції з іншими системами.

Це дозволяє забезпечити гнучкість та адаптивність додатку, що є важливим для підтримки його актуальності та ефективності в умовах швидкозмінного технологічного середовища.

3.2.4 Продуктивність

Продуктивність додатку є критичним фактором для забезпечення позитивного досвіду користувачів. Час відповіді на більшість запитів становить 1-2 секунди, що забезпечує швидке реагування додатку на дії користувача. Це досягається за рахунок оптимізації коду та ефективного використання ресурсів пристрою.

Додаток здатний справлятися зі збільшеним навантаженням шляхом масштабування. Вертикальне масштабування дозволяє збільшувати потужність окремих компонентів системи, а горизонтальне масштабування забезпечує додавання нових екземплярів серверів для розподілу навантаження.

Для забезпечення стабільної роботи додатку в умовах високого навантаження використовуються сучасні технології кешування та оптимізації запитів до бази даних. Це дозволяє знизити навантаження на сервер та покращити загальну продуктивність системи.

3.2.5 Вимоги до бази даних

У контексті розробки мобільного додатку класифікатора зображень на основі штучного інтелекту важливим є рішення щодо використання або невикористання бази даних. В даному додатку база даних не реалізована, і таке рішення є доцільним з кількох причин. По-перше, незбереження стану між сесіями та локальне зберігання даних дозволяють додатку виконувати класифікацію зображень без необхідності зберігати інформацію про попередні сесії або результати класифікації, обробляючи і виводячи всі дані в реальному часі без збереження результатів у довгострокову пам’ять. Це забезпечує простоту та швидкість роботи додатку, оскільки відсутність

бази даних уникає зайвої складності, що може виникнути при її інтеграції та управлінні, спрощуючи архітектуру та підтримку додатку. Відсутність операцій з читанням/записом до бази даних забезпечує більш швидкий відгук додатку, оскільки всі операції з даними виконуються безпосередньо в пам’яті пристрою.

3.2.6 Інші вимоги

Для класифікації зображень використовується TensorFlow Lite, інтегрований у мобільний додаток. TensorFlow Lite забезпечує ефективну роботу моделей машинного навчання на мобільних пристроях з обмеженими ресурсами. Необхідне підключення до Інтернету для завантаження та обробки зображень.

ДОДАТОК Ж

Слайди презентації

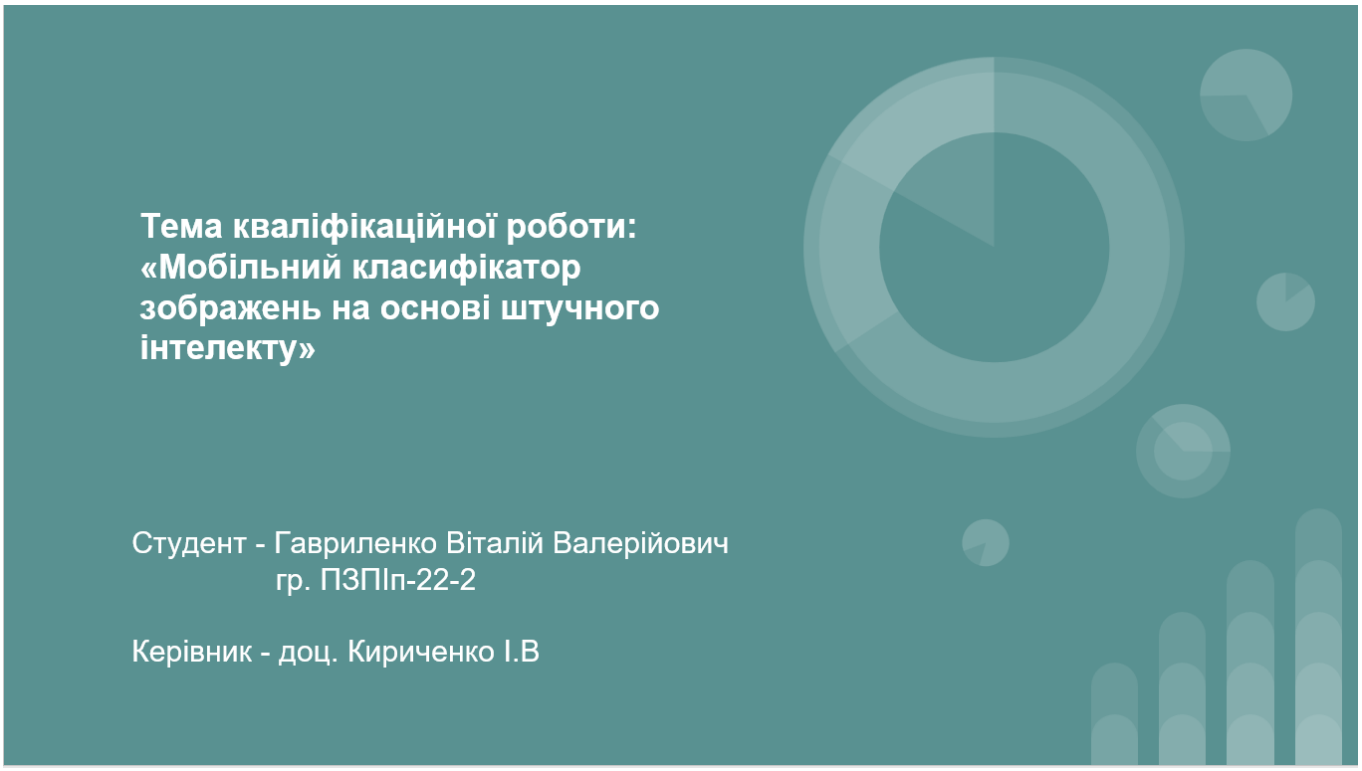


Рисунок Ж.1 – Слайд 1 (виконано самостійно)

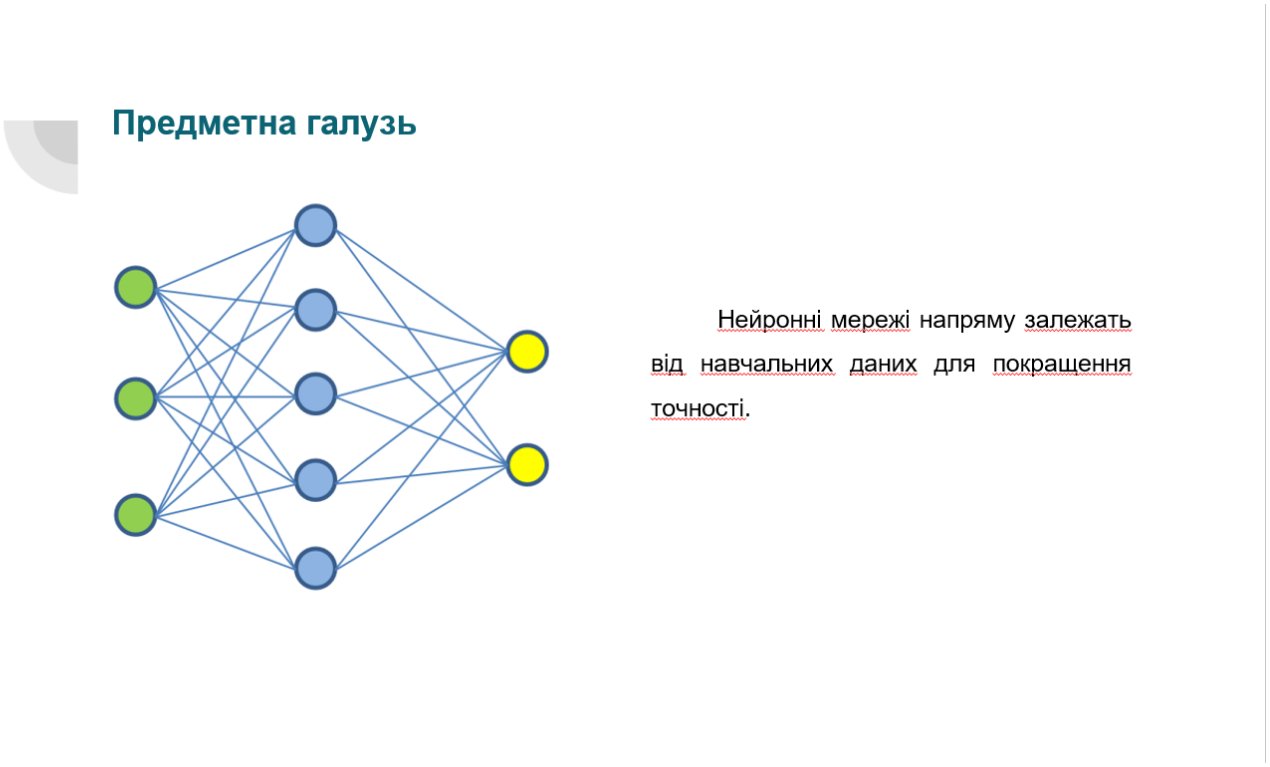


Рисунок Ж.2 – Слайд 2 (виконано самостійно)

Рисунок Ж.3 – Слайд 3 (виконано самостійно)

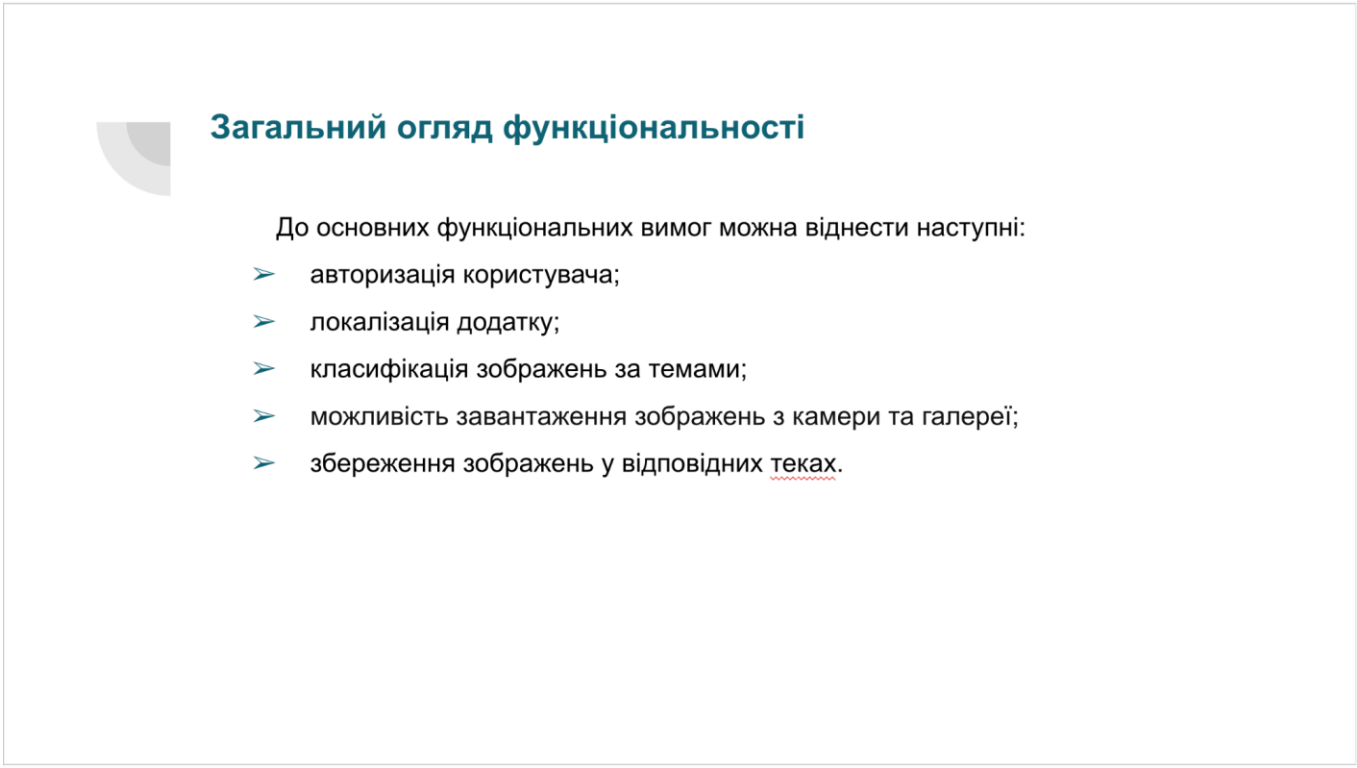


Рисунок Ж.4 – Слайд 4 (виконано самостійно)

Порівняння аналогів

Параметр	Google Lens	Microsoft Seeing AI	CamFind
Точність розпізнавання	Висока	Висока	Висока
Інтернет-залежність	Потрібен	Потрібен	Потрібен
Цільова аудиторія	Загальна	Люди з вадами зору	Загальна
Підтримка реального часу	Так	Так	Так
Підтримка багатьох мов	Так	Так	Ні
Конфіденційність даних	Питання конфіденційності	Питання конфіденційності	Питання конфіденційності
Функціональність	Широка	Спеціалізована для допомоги	Універсальна

Рисунок Ж.5 – Слайд 5 (виконано самостійно)

Використані технології

- TensorFlow Lite;
- Java;
- Firebase;
- Android SDK.

Рисунок Ж.6 – Слайд 6 (виконано самостійно)

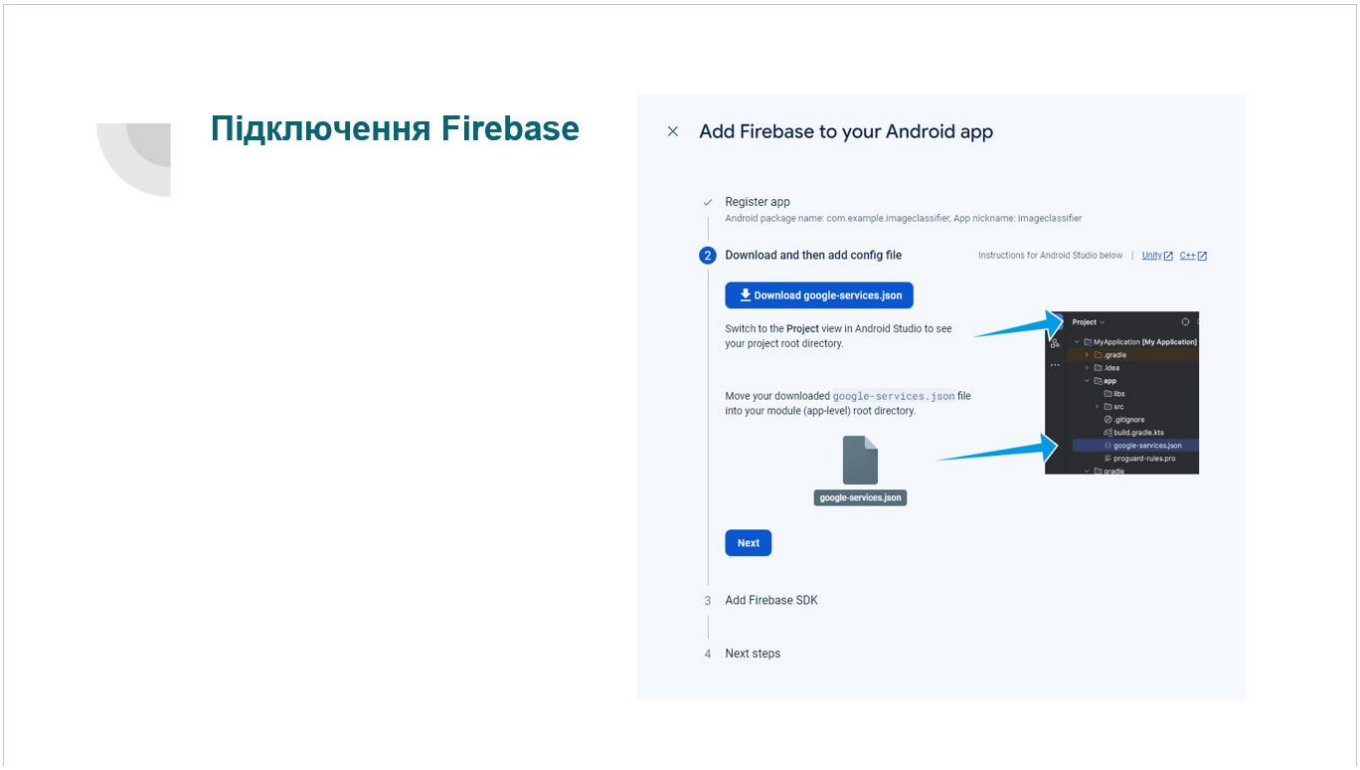


Рисунок Ж.7 – Слайд 7 (виконано самостійно)

Рисунок Ж.8 – Слайд 8 (виконано самостійно)

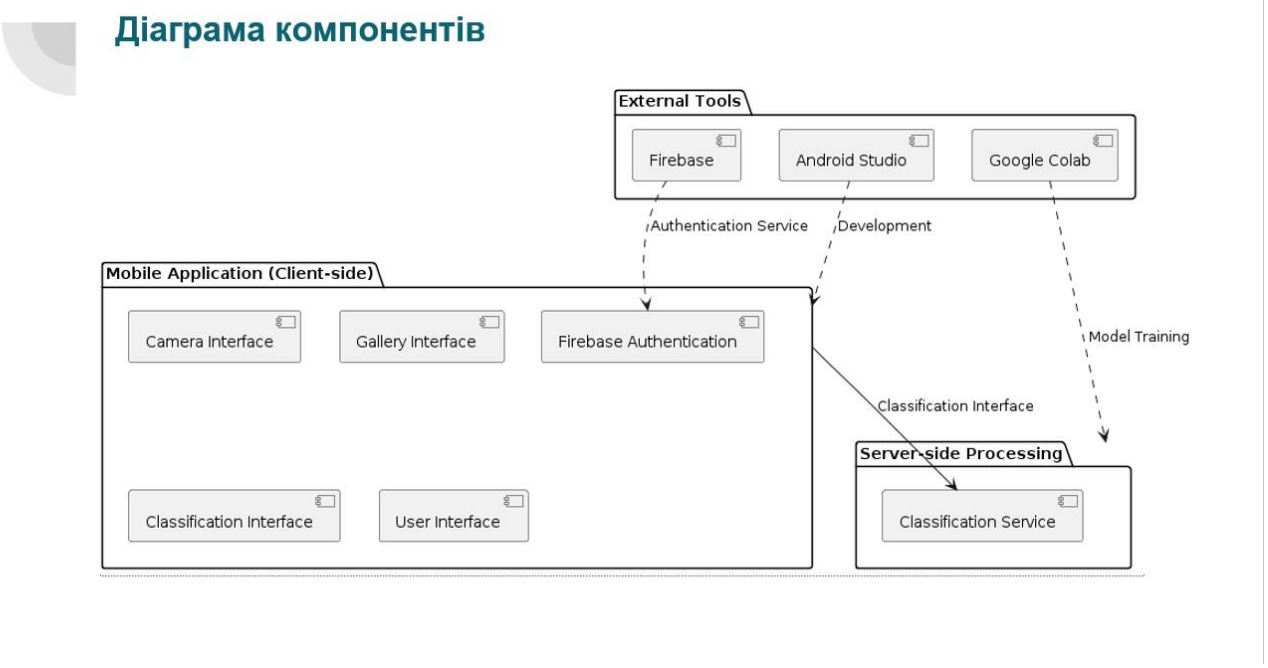


Рисунок Ж.9 – Слайд 9 (виконано самостійно)

Рисунок Ж.10 – Слайд 10 (виконано самостійно)

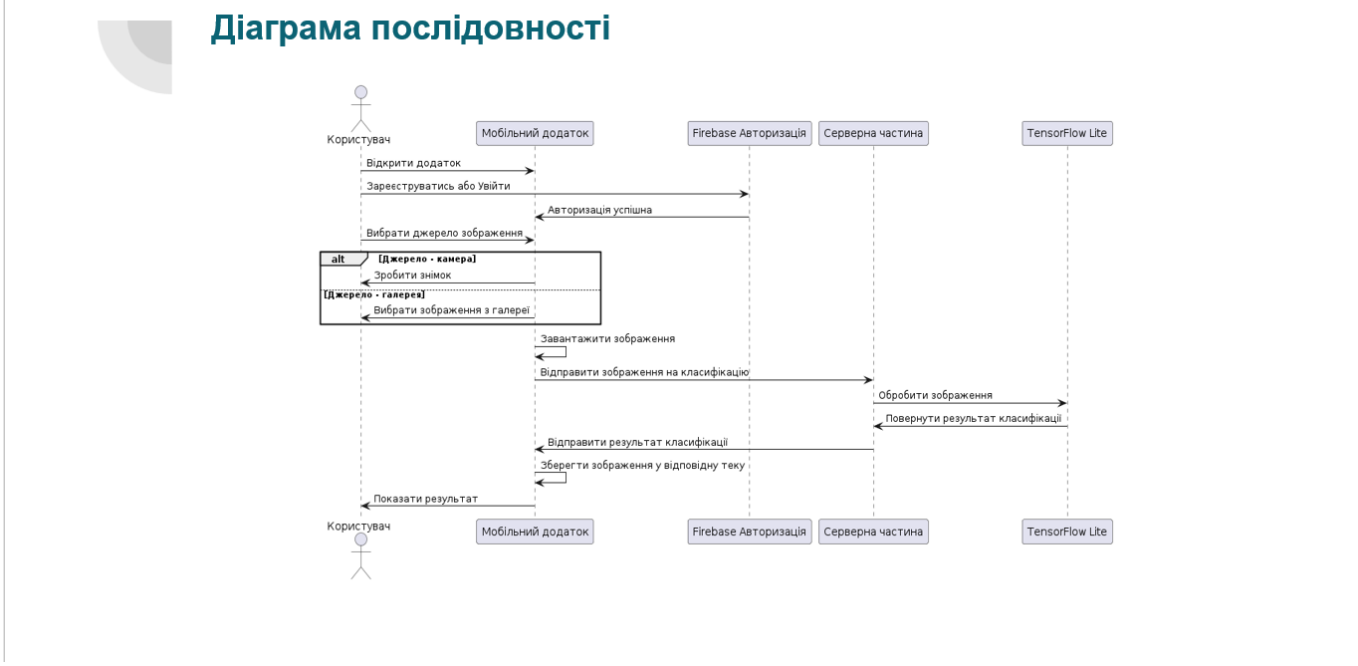


Рисунок Ж.11 – Слайд 11 (виконано самостійно)

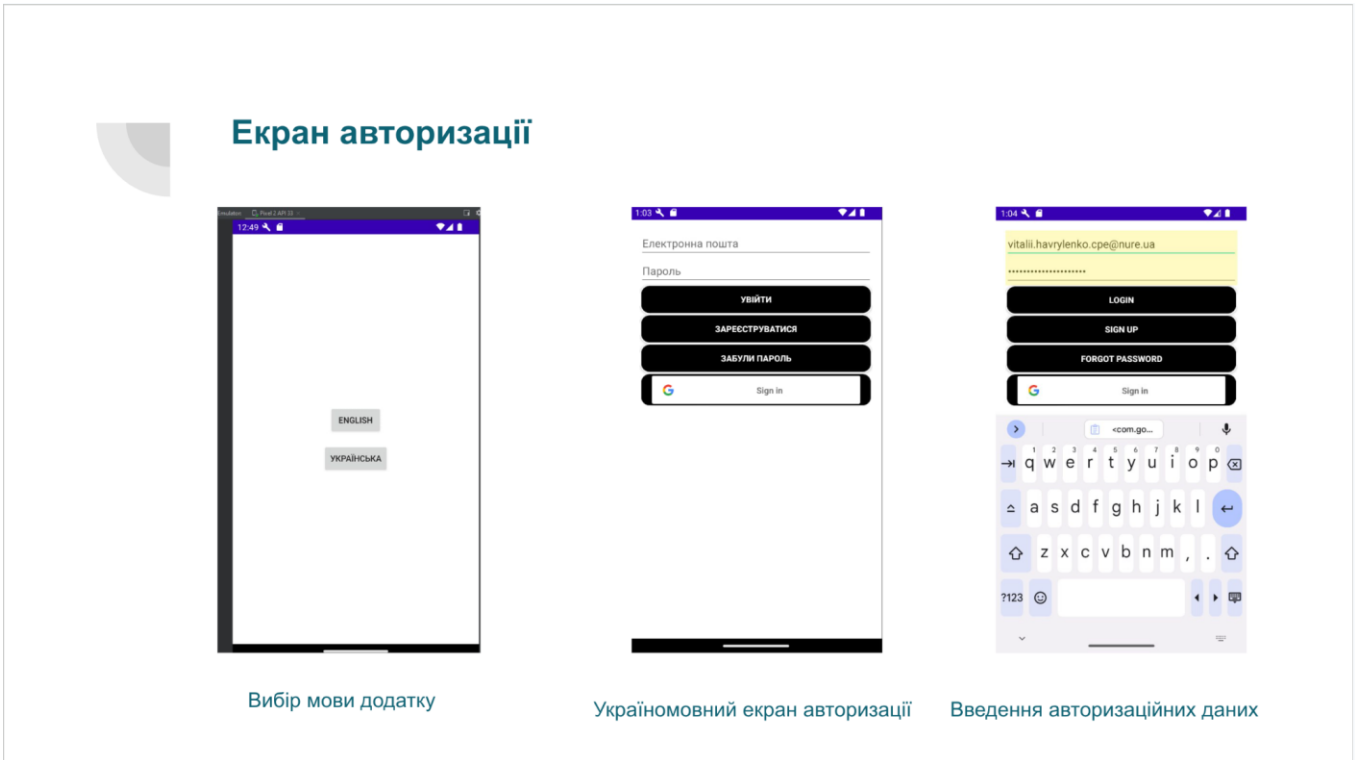


Рисунок Ж.12 – Слайд 12 (виконано самостійно)

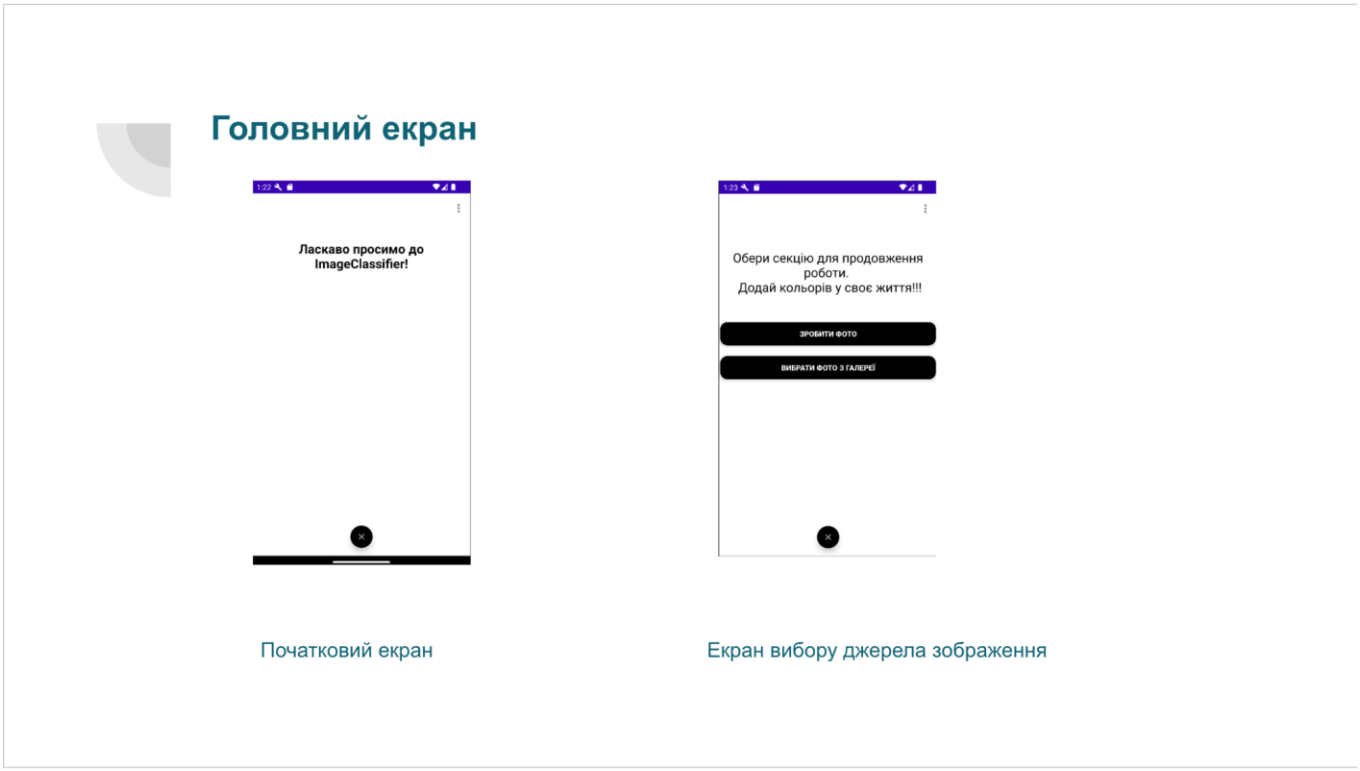


Рисунок Ж.13 – Слайд 13 (виконано самостійно)

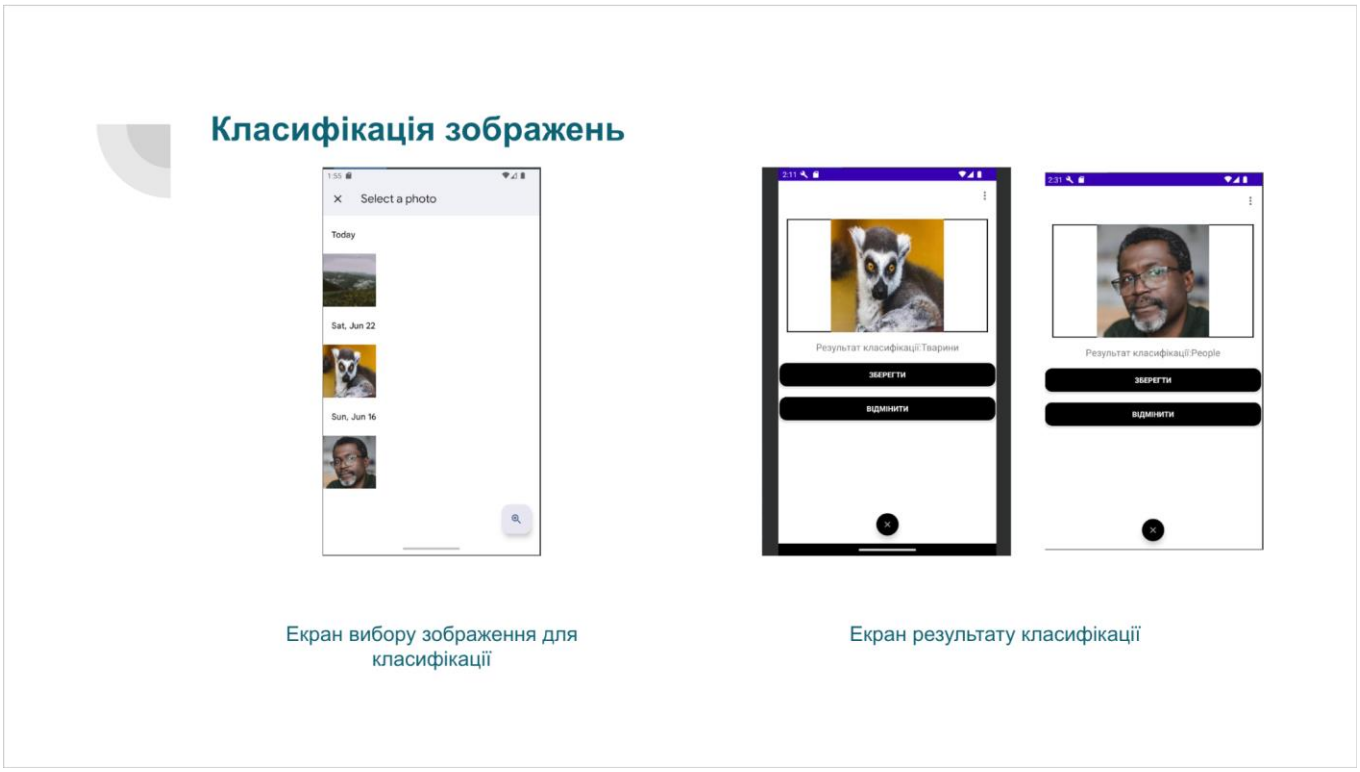


Рисунок Ж.14 – Слайд 14 (виконано самостійно)

Тест-кейс “Перевірка реєстрації нового користувача”

ID кейсу	1
Назва кейсу	Перевірка реєстрації нового користувача
Опис	Перевірка можливості реєстрації нового користувача з валідними даними
Передумови	Додаток запущений, користувач знаходиться на екрані реєстрації
Кроки	1. Ввести валідну адресу електронної пошти 2. Ввести валідний пароль 3. Підтвердити пароль 4. Натиснути кнопку 'Зареєструватися'
Очікуваний результат	Користувач успішно зареєстрований, перенаправлений на головний екран

Рисунок Ж.15 – Слайд 15 (виконано самостійно)

Тест-кейс “Перевірка відновлення паролю”

ID кейсу	7
Назва кейсу	Перевірка відновлення паролю
Опис	Перевірка функціоналу відновлення паролю через електронну пошту
Передумови	Додаток запущений, користувач знаходиться на екрані відновлення паролю
Кроки	1. Ввести зареєстровану адресу електронної пошти 2. Натиснути кнопку 'Відновити пароль' 3. Перевірити електронну пошту на наявність листа з інструкціями
Очікуваний результат	Користувач отримує лист з інструкціями для відновлення паролю

Рисунок Ж.16 – Слайд 16 (виконано самостійно)

Тест-кейс “Перевірка відповідності класифікації”

ID кейсу	15
Назва кейсу	Перевірка відповідності класифікації
Опис	Перевірка відповідності класифікації обраного зображення за темою - тварини
Передумови	Додаток запущений, користувач обрав фото з файлового сховища пристрою за темою тварини
Кроки	1. Обрати фото за темою 2. Натиснути на обране фото 3. Перевірити результат класифікації
Очікуваний результат	Після обрання фотографії користувачу відображається обране фото з темою класифікації під ним. Кнопки зберегти та відмінити присутні на формі.

Рисунок Ж.17 – Слайд 17 (виконано самостійно)

Виклики та рішення

- інтеграція різних технологій та сервісів (Firebase, TensorFlow Lite);
- забезпечення швидкої та точної класифікації зображень;
- великі потужності та об’єми даних для навчання моделі

Рисунок Ж.18 – Слайд 18 (виконано самостійно)