НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмний застосунок для виявлення логотипів брендів на фото на основі штучного інтелекту

Виконав студент IV курсу, групи ІТ-01
(шифр групи)

Терешпольський Антон Ігорович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., Новінський В. П. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., Солдатова М. О. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2024

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Терешпольському Антону Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Застосунок для виявлення логотипів брендів на фото на основі штучного інтелекту

керівник проєкту Новінський Валерій Петрович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «27» травня 2024 р. №2112-с

2. Термін подання студентом проєкту « 17 » червня 2024 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.
4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань
2) Схема бізнес процесів застосунку
3) Архітектура програмного забезпечення
4) Графіки параметрів моделі
5) Матриця плутанини моделі
6) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «11» березня 2024 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	11.03.2024	
2	Аналіз існуючих методів розв’язання задачі	16.03.2024	
3	Постановка та формалізація задачі	26.03.2024	
4	Розробка інформаційного забезпечення	11.04.2024	
5	Алгоритмізація задачі	16.04.2024	
6	Обґрунтування вибору використаних технічних засобів	25.04.2024	
7	Розробка програмного забезпечення	13.05.2024	
8	Налагодження програми	16.05.2024	
9	Виконання графічних документів	20.05.2024	
10	Оформлення пояснювальної записки	07.06.2024	
11	Подання ДП на попередній захист	18.05.2024	
12	Подання ДП рецензенту	07.06.2024	
13	Подання ДП на основний захист	14.06.2024	

Студент _____ Антон ТЕРЕШПОЛЬСЬКИЙ
(підпис) (ініціали, прізвище)

Керівник _____ Валерій НОВІНСЬКИЙ
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з п’яти розділів, містить 31 таблицю, 19 рисунків та 5 джерел – загалом 44 сторінки.

Дипломний проєкт присвячений розробці застосунку для знаходження логотипів на фото на основі штучного інтелекту.

Мета дипломного проєкту полягає в автоматизації виявлення та розмиття логотипів 5 різних брендів на фото, за допомогою застосунку з простим інтерфейсом, для уникнення проблем з авторським правом.

Об'єкт дослідження: застосунок «Logo Detector» для виявлення логотипів брендів на фото на основі штучного інтелекту.

Предмет дослідження: технологія машинного навчання як засіб для автоматизації задач обробки фото.

У першому розділі проведено аналіз предметної області та вимог до програмного забезпечення. Сформульовано функціональні та нефункціональні вимоги.

Другий розділ присвячений огляду інструментів розробки. Розглянуто особливості розробки застосунків, що використовують штучний інтелект.

У третьому розділі описано моделювання та конструювання програмного забезпечення. Розроблено архітектуру. Спроектовано інтерфейс і навчено модель для розпізнавання логотипів. Реалізовано накладання ефекту розмиття.

У четвертому розділі було проведено аналіз якості і тестування програмного забезпечення.

П’ятий розділ присвячено впровадженню та супроводу створеного програмного застосунку.

Результатом роботи є застосунок, що надає можливість користувачу в автоматичному режимі накладати ефект розмиття на логотипи, на фотографіях.

КЛЮЧОВІ СЛОВА: ДОДАТОК, LOGO, WINDOWS, DETECTION, PYTHON, PYCHARM, YOLO, МОДЕЛЬ, ФОТО.

ABSTRACT

The explanatory note of the diploma project consists of five chapters, includes 31 tables, 19 figures, and 5 sources, totaling 44 pages.

The diploma project is dedicated to the development of an application for finding logos in photos based on artificial intelligence.

The aim of the diploma project is to create an application with a simple interface for automating the detection and blurring of logos of 5 different brands in photos to avoid copyright issues.

Object of research: the "Logo Detector" application for detecting brand logos in photos based on artificial intelligence.

Subject of research: machine learning technology as a means for automating photo processing tasks.

The first chapter analyzes the subject area and software requirements. Functional and non-functional requirements are formulated.

The second chapter is devoted to the review of development tools. The features of developing applications that use artificial intelligence are considered.

The third chapter describes the modeling and design of the software. The architecture is developed. The interface is designed and the model for logo recognition is trained. The blurring effect is implemented.

In the fourth chapter, the quality analysis and software testing were conducted.

The fifth chapter is dedicated to the implementation and maintenance of the created software application.

The result of the work is an application that allows the user to automatically apply a blurring effect to logos in photos.

KEYWORDS: APPLICATION, LOGO, WINDOWS, DETECTION, PYTHON, PYCHARM, YOLO, MODEL, PHOTO.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-0125.045480.01.91	Технічне завдання	12	
3	А4	КПІ.ІП-0125.045480.02.81	Пояснювальна записка	44	
4	А4	КПІ.ІП-0125.045480.03.12	Текст програми	10	
5	А4	КПІ.ІП-0125.045480.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІП-0125.045480.05.34	Керівництво користувача	7	
7	А3	КПІ.ІП-0125.045480.06.99	Графічний матеріал	7	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІТ-0125.045480.00.90		
Розробн.		Терешпольський А.І.			Відомість дипломної роботи	Літ.	Лист 1 / Листів 1
Керівник.		Новінський В. П.					
Консульт						КПІ ім.Ігоря Сікорського кафедра ІПІ гр. ІТ-01	
Н/контр.		Вітковська І. І.					
зав.каф.		Жаріков Е. В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2024 р.

ЗАСТОСУНОК ДЛЯ ВИЯВЛЕННЯ ЛОГОТИПІВ БРЕНДІВ НА ФОТО НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

Технічне завдання

КПІ.ІТ-0125.045480.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Валерій НОВІНСЬКИЙ

Нормоконтроль:

_____ Ірина ВІТКОВСЬКА

Виконавець:

_____ Антон ТЕРЕШПОЛЬСЬКИЙ

Київ – 2024

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки:

Застосунок для виявлення логотипів брендів на фото на основі штучного інтелекту.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного застосунку «Logo Detector» [045480], котрий використовується для виявлення та маскування логотипів на фотографіях та призначений для галузі фотозйомки, для спрощення обробки фото і уникнення проблем з авторським правом.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програмного застосунку «Logo Detector» є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для спрощення та автоматизації обробки фото і приведення їх до можливості комерційної публікації шляхом розмиття логотипів брендів. Це сприятиме уникненню проблем з авторським правом при комерційному використанні фотографій.

Метою розробки є автоматизація виявлення та розмиття логотипів 5 різних брендів на фото, за допомогою застосунку з простим інтерфейсом.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

– вікно входу в профіль користувача (рисунок 4.1);

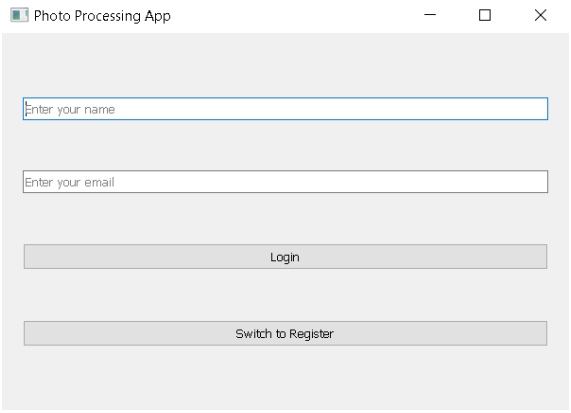


Рисунок 4.1 – Вікно входу

– вікно реєстрації в додатку (рисунок 4.2);

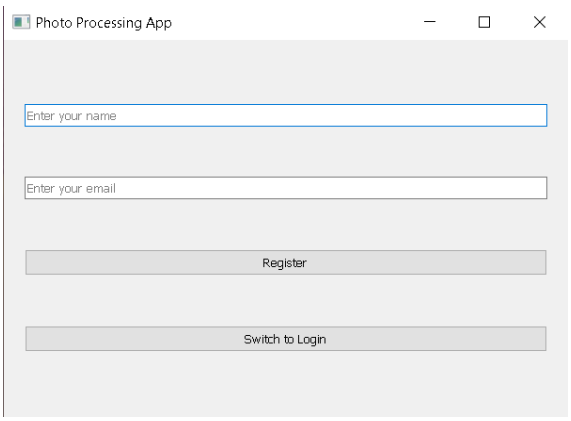


Рисунок 4.2 – Вікно реєстрації

– вікно профіля користувача (рисунок 4.3);

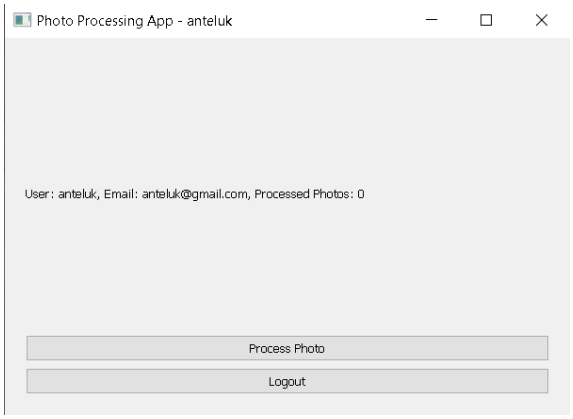


Рисунок 4.3 – Вікно профілю

4.1.2 Для користувача:

– функція реєстрації;
– функція входу до профіля;
– функція виходу з профіля;
– функція виявлення логотипів на фото;
– функція виявлення бренду до якого відноситься логотип;
– функція накладання ефекту розмиття;
– функція підрахунку оброблених фото.

4.1.3 Для адміністратора системи (якщо він передбачений):

– адміністратор не передбачений.

4.1.4 Додаткові вимоги:

– програма має мати доступ до файлової системи комп’ютера.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача.

Так як ПЗ являє собою невеликий додаток для ПК на базі ОС Windows, вимоги до надійності мінімальні. Додаток має працювати без збоїв, займати

невелику кількість пам’яті. При можливому збою, перезавантажити або перевстановити додаток.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core 2 Duo або аналоги від AMD;
– об‘єм ОЗП: 4 Гб;
– Windows 10 64-біт, Windows 8 64-біт.

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i3/i5/i7 або аналоги від AMD;
– об‘єм ОЗП: 8 Гб;
– Windows 10 64-біт, Windows 8 64-біт.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN64.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі:

– .png;
– .jpg;
– .jpeg;
– .bpm.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі:
– .png;
– .jpg;
– .jpeg;
– .bpm.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Python.

4.5.4 Вимоги до середовища розробки

Розробку виконати за допомогою середовища PyCharm.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді текстових додатків.

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема бізнес процесів застосунку;
– схема структурна варіантів використання;
– архітектура програмного забезпечення та моделі;
– креслення вигляду екранних форм;
– матриця плутанини моделі;
– графіки параметрів моделі.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	11.03	
2.	Розробка технічного завдання	18.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	27.03	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	05.04	Схема структурна програмного забезпечення та специфікація компонентів
5.	Програмна реалізація програмного забезпечення	13.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	16.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	20.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	25.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	06.06	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: **Застосунок для виявлення логотипів брендів на фото на основі штучного інтелекту**

КПІ.ІТ-0125.045480.02.81

Київ – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IDE	–	Integrated Development Environment – інтегроване середовище розробки.
YOLO	–	You Only Look Ones
ШІ	–	Штучний Інтелект
ІТ	–	Інформаційні технології
BPMN	–	Business process model and notation
ОС	–	Операційна система.

ВСТУП

В наші дні велику популярність набирають соціальні мережі такі як «Instagram», «YouTube», «TikTok» і тд. Ці соцмережі передбачають активне створення медіа контенту. Користувачі активно створюють контент яким діляться на велику аудиторію, а дехто навіть на цьому заробляє.

Застосунок «Logo Detector» створювався з розрахунком саме на тих контентмейкерів, які на своєму контенті заробляють. Це можуть бути фотографи або моделі які продають свої фото, або заробляють іншим чином використовуючи зроблені фото в публічній площині. Дуже часто на фотографіях може з’явитись контент, захищений авторським правом. Це може призвести до небажаних ситуацій і проблем з законодавством. Багато контентмейкерів робить фото на вулицях міста і на фото потрапляє багато логотипів різних брендів. Фотографам доводиться замальовувати їх вручну.

Додаток допоможе автоматизувати виявлення і розмиття ділянки з логотипом певного бренду. Це спростить обробку фотографій, адже тепер контентмейкеру не потрібно знаходити на своєму фото логотип, встановлювати стороннє ПЗ для того щоб цей логотип розмити, шукати в інтернеті правову інформацію.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Компанії та організації, такі як Statista, Pew Research Center та інші, регулярно випускають звіти та статистику щодо використання соціальних мереж. З кожним роком кількість користувачів росте (Рис 1.1.1). За даними дослідження, в 2023 році, кількість аккаунтів вже майже досягла 5 мільярдів [5].

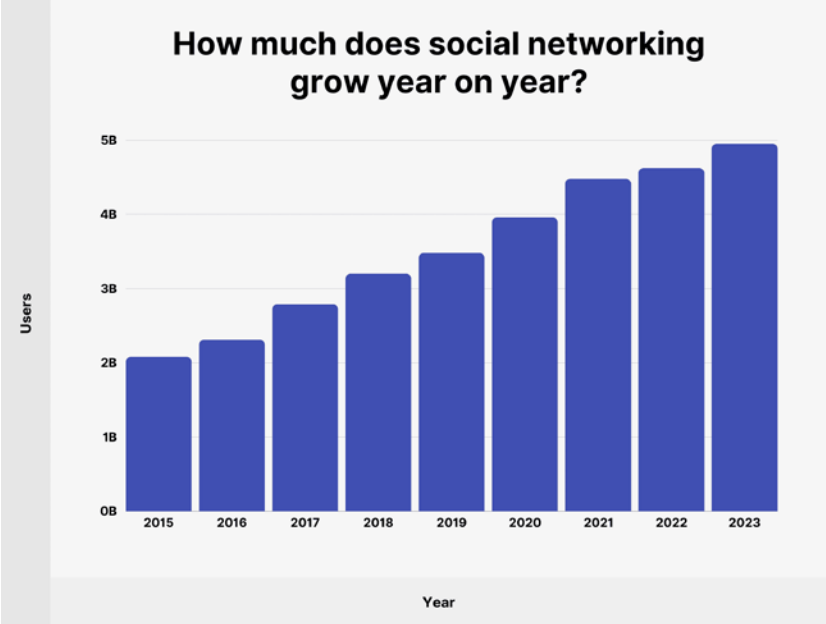


Рисунок 1.1.1 – Зростання використання соцмереж

Популярність соціальних мереж обумовлена їхнім важливим внеском у збереження зв'язку та спілкування, можливістю виражати себе через візуальний контент, задоволенням інформаційних потреб, використанням для брендування та маркетингу, а також сприянням, підтримкою та обговоренням важливих тем. Ці мережі стали не тільки засобом знаходження контенту, а й платформою для спілкування, обміну думками та виявлення особистої ідентичності, завдяки інноваціям та зручності використання в онлайн-середовищі.

Соціальні мережі стали невід'ємною платформою для медіакреаторів, які активно використовують цей канал для заробітку. Медіакреатори створюють та публікують унікальний контент, будь то відео, фотографії чи блоги,

залучаючи велику аудиторію. Вони монетизують свою активність через різноманітні методи, такі як рекламні співпраці, партнерські програми, або отримання доходу від підписників чи фанатів через платформи підписки. Сприятливе середовище соціальних мереж дозволяє медіакреаторам не лише виражати свою творчість, але й отримувати прибуток від свого контенту, що робить їхню діяльність важливою для економіки в онлайн-середовищі.

Мати програму для редагування фотографій під рукою є важливим аспектом в сучасному світі, оскільки це надає можливість ефективно працювати з графікою та забезпечує контроль за власним образом в онлайн-середовищі. Редагування фото може допомогти уникнути потенційних проблем з авторським правом, оскільки ви можете видаляти або модифікувати відзначення авторства та інші ідентифікуючі елементи. Крім того, це дає можливість коригувати зображення з точки зору естетики та зберігати конфіденційність, адаптуючи фотографії відповідно до особистих потреб і вимог соціальних мереж чи платформ. Наявність такого інструменту полегшує взаємодію в онлайн-середовищі, забезпечуючи більший контроль та творчу свободу.

Використання автоматизованих програм для розмиття логотипів стає надзвичайно зручною альтернативою в порівнянні з ручними методами. Замість того, щоб витрачати час на пошук логотипів, завантаження сторонніх програм для обробки фотографій та вручне закривати логотипи, автоматизовані інструменти надають швидке та ефективне рішення. Вони пропонують простий інтерфейс, що дозволяє за кілька кроків розмити логотипи на зображеннях без необхідності складних процедур. Це зберігає час та витрати енергії, роблячи процес редагування фотографій більш ефективним та зручним для користувачів. Такий підхід сприяє автоматизації та полегшує використання ресурсів для тих, хто шукає ефективні рішення в обробці зображень.

Використання нейронних мереж для детекції логотипів на зображенні є обґрунтованим та ефективним підходом у сучасному комп'ютерному зорі.

Нейронні мережі, здатні автоматично розпізнавати та визначати положення об'єктів, у тому числі логотипів, на зображеннях. Вони володіють великою вмісткістю та можливістю самостійного навчання на великих обсягах даних, що дозволяє їм адаптуватися до різноманітних контекстів та стилів зображень. Завдяки глибокому навчанню, нейронні мережі можуть автоматично виявляти логотипи з високою точністю та швидкістю, що робить їх незамінним інструментом для завдань детекції об'єктів на великому масштабі в сучасній обробці зображень..

Для якісної обробки фотографій ключовим фактором є використання персонального комп'ютера (ПК). ПК надає великі можливості щодо обчислень та обробки графіки, що дозволяє використовувати потужні графічні редактори та спеціалізовані програми. Тому в якості платформи для якої розроблявся додаток була вибрана Windows.

При розробці застосунків, особливо тих, які спрямовані на обробку фотографій або інші творчі завдання, не завжди ставляться високі вимоги до користувацького інтерфейсу. Важливим є те, щоб інтерфейс був простим, логічним та виконував основні завдання без зайвих складнощів. Користувачам зазвичай потрібен швидкий та інтуітивно зрозумілий доступ до основних функцій, таких як завантаження, редагування та збереження зображень. Спрощений інтерфейс дозволяє користувачам зосередитися на творчому процесі та ефективно використовувати функціонал програми, мінімізуючи витрати часу та зусиль на освоєння складних операцій. Такий підхід сприяє зручності використання застосунку та робить його доступним для широкого кола користувачів.

Додавання функціоналу для відстеження та відображення кількості оброблених фотографій може стати важливим елементом користувацького досвіду. Це не лише надасть користувачеві можливість стежити за своєю продуктивністю та досягненнями, але і стимулюватиме його активність та зацікавленість у використанні програми. Знання кількості оброблених фотографій за час користування може слугувати внутрішньою метрикою

успіху, а також створювати додатковий стимул для подальшого використання застосунку. Ця функція сприяє залученню користувача та виробленню позитивного враження від використання програми для обробки фотографій.

Для розробки застосунку «Logo Detector» було вирішено використовувати мову Python. Саме ця мова є стандартною коли йдеться про додатки з ШІ. Для навчання нейронної мережі був сформований та розмічений датасет, що містить в собі 201 зображення 5 відомих брендів. Для розмітки датасету використовувалася безкоштовна версія застосунку CVAT. Також в якості архітектури нейронної мережі була використана архітектура You Only Look Once v8 (YOLOv8).

1.2 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації застосунку «Logo Detector». Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

1.2.1 Аналіз відомих програмних продуктів

Існує багато відомих редакторів фотографій. Наприклад Adobe Photoshop (Рис. 1.2.1.1). В цьому застосунку є широкий спектр інструментів для редагування фото (Рис. 1.2.1.2). Але на відміну від застосунку «Logo Detector» він не має функції автоматизованого розпізнавання логотипів (на момент створення додатку). Тому можна використовувати його в парі з розробленим застосунком для вирішення більш складних задач.

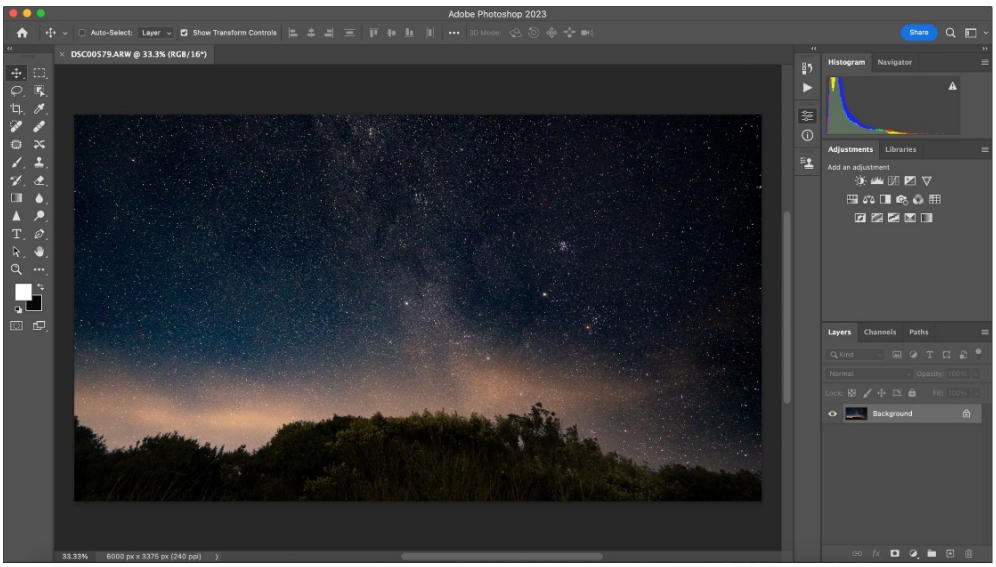


Рисунок 1.2.1.1 – Вигляд основного вікна застосунку «Adobe Photoshop»

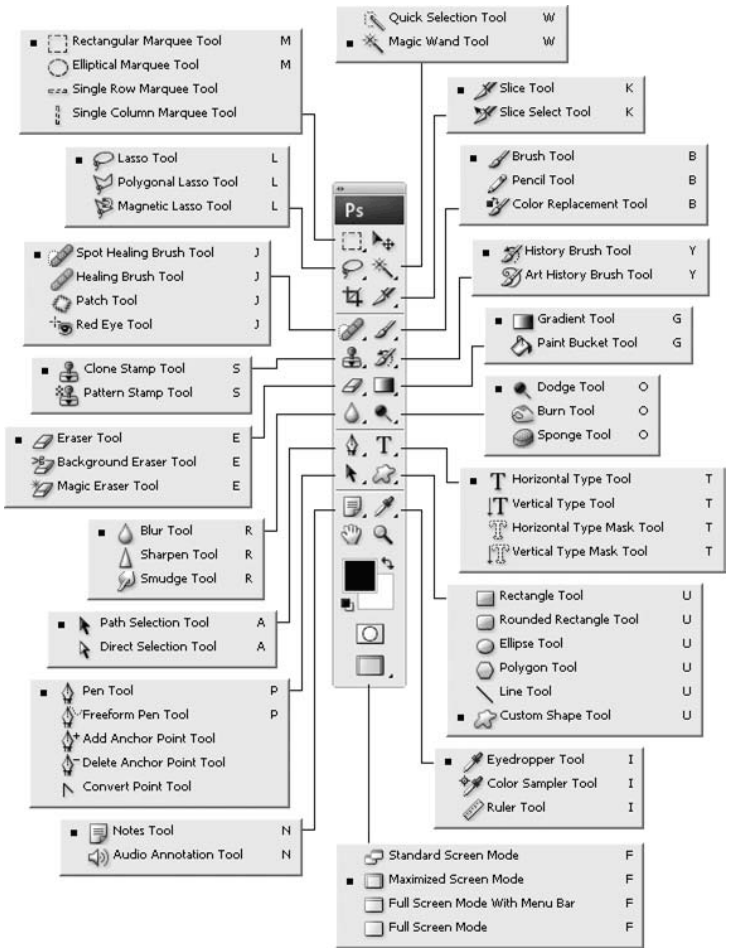


Рисунок 1.2.1.2 – інструменти застосунку «Adobe Photoshop»

Також існують такі редактори як : GIMP, Canva (Рис. 1.2.1.3 – 1.2.1.4).

В порівнянні з ними, моя програма, розроблена з використанням PyQt5 та інших бібліотек Python, робить акцент на свою спеціалізацію в області

обробки фотографій. Забезпечуючи можливість об'єднання облікового запису, обробки зображень та застосування алгоритмів виявлення об'єктів, програма відзначається розширеними функціональними можливостями. Це особливо цінно для користувачів, які шукають інструменти з додатковими можливостями у сфері обробки зображень та мають конкретні потреби у фотографії. Завдяки використанню бібліотек Python, програма дозволяє ефективно взаємодіяти з обраним обладнанням та оптимізує робочий процес обробки фотоматеріалів.

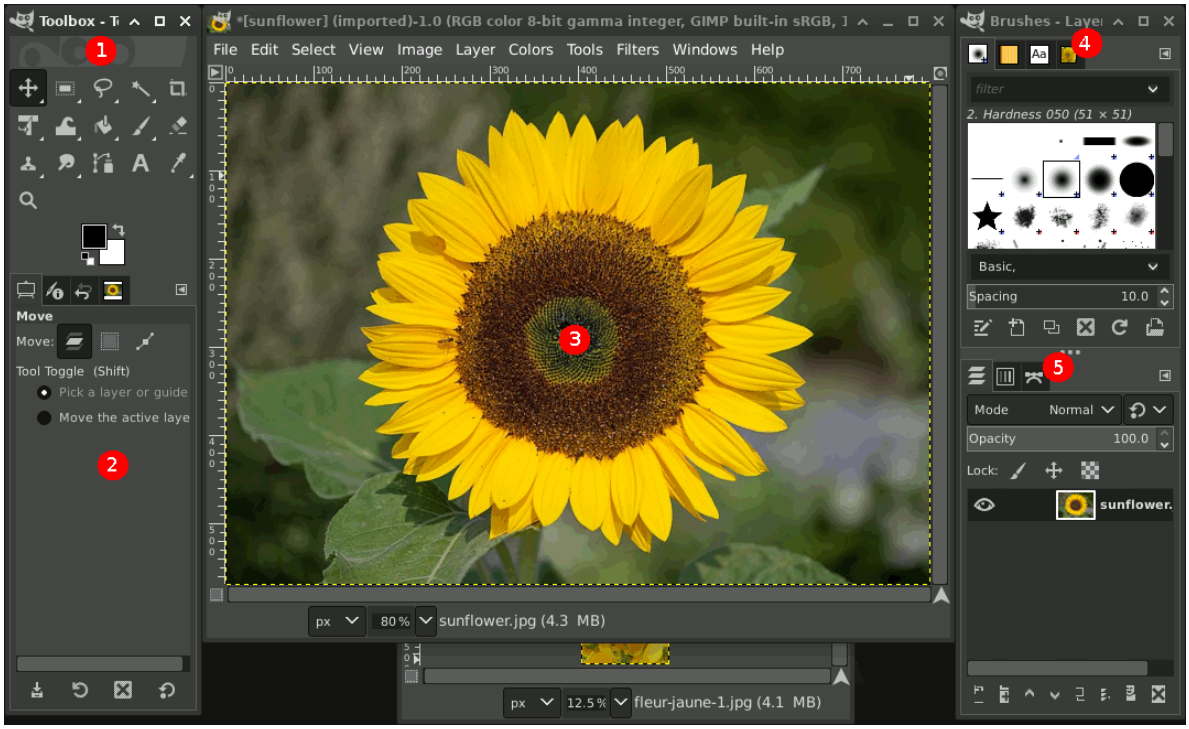


Рисунок 1.2.1.3 – Вигляд застосунку «GIMP»

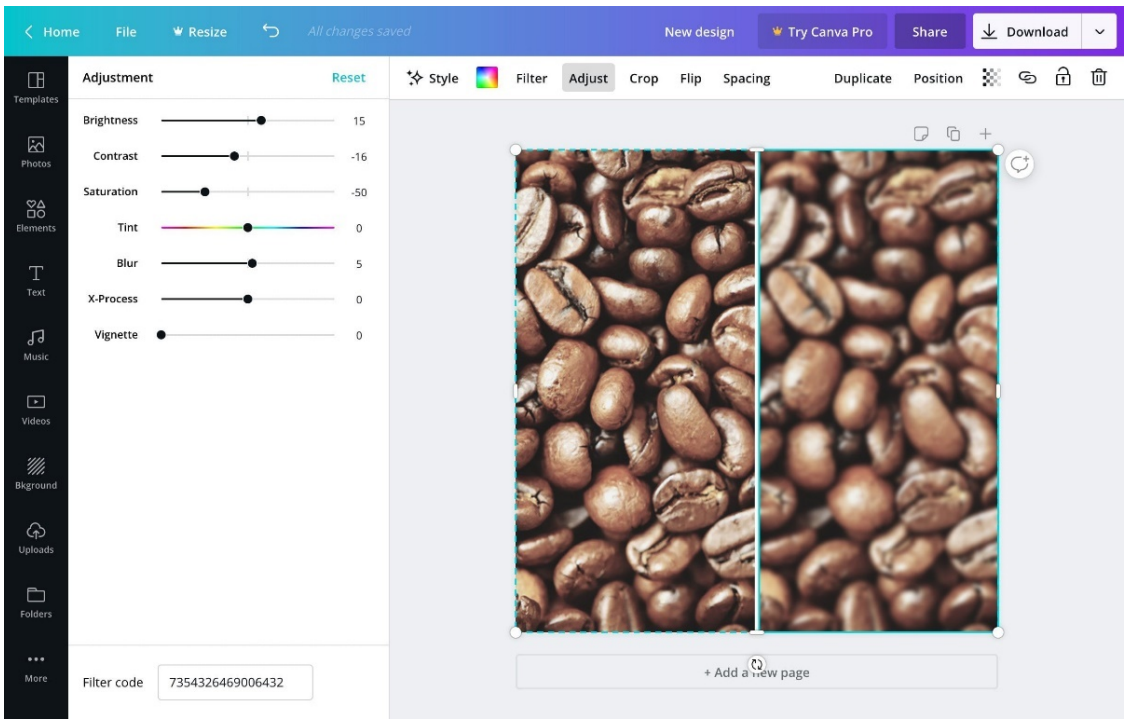


Рисунок 1.2.1.4 – Вигляд застосунку «Canva»

В таблиці 1.2.1.1 порівняно функціональні можливості застосунків «Logo Detector», «Canva» та «GIMP».

Таблиця 1.2.1.1 – Порівняння з аналогом

Функціонал	GIMP	Canva	Logo Detector
Реєстрація користувача	Реєстрація не передбачена	Реєстрація присутня	Реєстрація присутня
Можливість відслідковувати кількість оброблених фото	Можливість не передбачена	Додаток дозволяє вести статистику проектів і повертатися до них пізніше	Додаток передбачає лічильник з кількістю оброблених фото
Можливість детекції об’єктів на фото	Можливість не передбачена	Можливість не передбачена	Додаток вміє знаходити на зображеннях логотипи брендів.

Продовження таблиці 1.2.1.1

Функціонал	GIMP	Canva	Logo Detector
Ефект розмиття	Є ручний інструмент для розмиття	Можливість не передбачена	Розмиття автоматизоване
Просунуті інструменти для обробки фото	Присутні інструменти для ретуші, корекції кольору, фільтри і тд.	Присутні тексти, шрифти, фільтри, ефекти, а також сумісна робота над фото	Можливість не передбачена

1.2.2 Аналіз відомих алгоритмічних та технічних рішень

У застосунку використовується алгоритм об’єктного виявлення на основі архітектури YOLO. Алгоритм об'єктного виділення представляє сучасний підхід до виявлення та локалізації об'єктів на зображеннях чи відео. Його основна особливість полягає в тому, що він розділяє зображення на сітку та одночасно проґрунтовує її, передбачаючи класи та координати об'єктів. Замість того, щоб використовувати складний процес виявлення об'єктів у кілька етапів, YOLO використовує одну нейронну мережу для здійснення цієї задачі за один прохід. Це забезпечує високу швидкість виявлення та високу точність результатів. YOLO широко використовується в застосунках комп'ютерного зору, автономних автомобілях, відеоспостереженні та інших областях, де важливо ефективно впоратися із завданнями об'єктного виявлення.

Програма включає у себе три екранні форми, які ретельно виважено для зручного та логічного взаємодії користувача з програмою. Форма логіну надає можливість введення ім'я та електронної пошти для входу в систему. Форма реєстрації дозволяє новим користувачам створити обліковий запис, вводячи своє ім'я та електронну адресу. На останній формі, яка є профільною,

користувач може переглядати свої особисті дані, обробляти фотографії та виконувати інші функції програми. Кожна форма ретельно розроблена з урахуванням простоти та логічності використання.

1.3 Опис бізнес-процесів

Для опису бізнес процесів програмного забезпечення використовується BPMN модель. На рисунку 1.3.1 представлено фрагмент моделі що демонструє процес авторизації в застосунку. Повна модель представлена на демонстраційному плакаті 1.

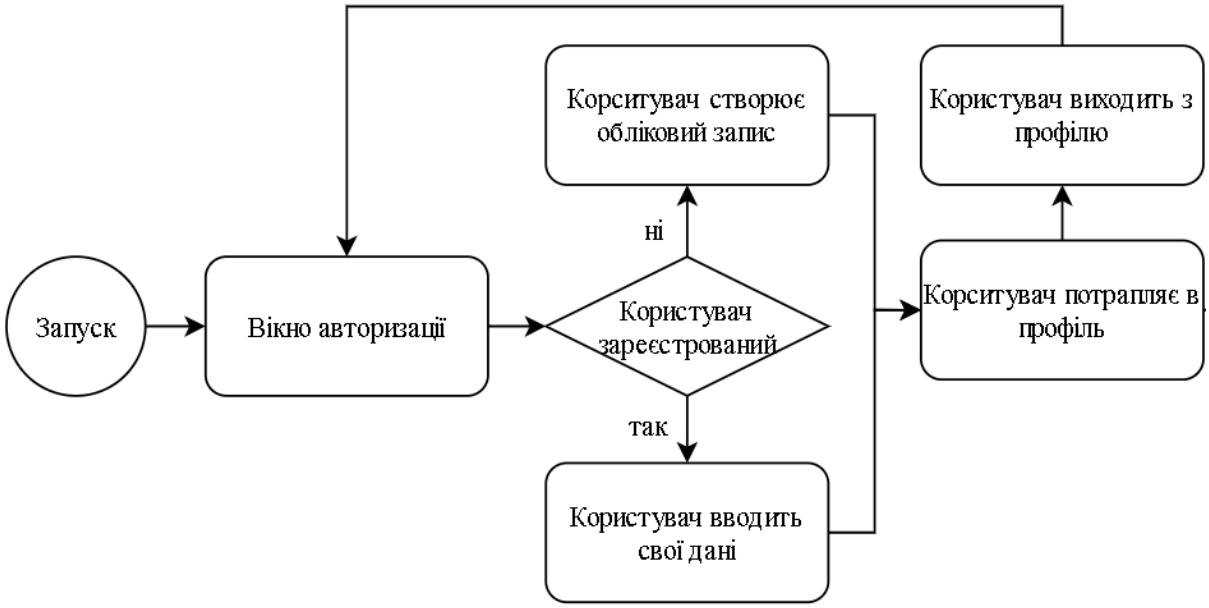


Рисунок 1.3.1– Процес авторизації

Опис послідовності роботи з застосунком:

– користувач відкриває застосунок;

– користувач авторизується;

– користувач переходить на екранну форму профілю;

– за потреби користувач переглядає лічильник і інформацію про себе;

– користувач натискає відповідну кнопку і обирає фото для обробки;

– користувач отримує результат обробки і інформацію про знайдені бренди;

– користувач зберігає оброблене фото.

1.4 Постановка задачі

Враховуючи всі описані вимоги, слід розробити програму для автоматичної детекції та розпізнавання логотипів на фотографіях з використанням технологій машинного навчання. Система повинна забезпечувати точну ідентифікацію логотипів навіть у складних умовах та в різних розмірах.

Важливою складовою є реалізація накладання ефекту розмиття на зображення. Саме це і є основною задачею застосунку. Детекція служить лише для того щоб мати координати куди накласти цей ефект, щоб логотипу було не видно на фото і контентмейкер міг без остраху його публікувати в своїх соцмережах.

Для детекції зображення слід розробити і навчити нейронну мережу на основі архітектури YOLO. Ця архітектура забезпечить надійну детекцію і типізацію логотипів на фотографіях.

Для початку треба буде сформувати датасет із зображеннями, а також розмітити його в безкоштовній версії CVAT. Після навчання нейронної мережі, за допомогою мови програмування Python та бібліотеки Ultralytics, розробити застосунок який би міг надати легкий та зрозумілий доступ до можливостей навченої нейромережі. Щоб кожен хто встановить програму і ознайомиться з нескладним інтерфейсом міг повною мірою користуватись цією розробкою.

Висновки до розділу

Отже, після аналізу предметної області та після аналізу застосунків, що є потенційними конкурентами було сформовано функціональні вимоги, які роблять застосунок «Logo Detector» гарним додатковим інструментом на

ринку застосунків для обробки фото. Адже більшість фоторедакторів не надають можливості автоматизувати пошук логотипів і їх розмиття.

Також було проаналізовано та обрано допоміжні програмні застосунки та бібліотеки що є найбільш ефективними для розробки саме цього застосунку.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Головною функцією програмного забезпечення є обробка фото шляхом знаходження та розмиття логотипа на фото, це можна побачити на рисунку 2.1.1. Повну діаграму варіантів використання представлено на кресленні 1.

Рисунок 2.1.1 – Фрагмент діаграми варіантів використання

В таблицях 2.1.1 - 2.1.8 наведені варіанти використання програмного забезпечення.

Таблиця 2.1.1 – Варіант використання UC-1

Use case name	Логін
Use case ID	UC-01
Goals	Авторизація в додатку
Actors	Користувач

Продовження таблиці 2.1.1

Trigger	Користувач бажає авторизуватися в додатку
Pre-conditions	В додатку ніхто не авторизований
Flow of Events	Користувач відкриває додаток та бачить вікно логіну. Користувач вводить свої дані, натискає кнопку і потрапляє в свій профіль
Extension	Якщо користувач ввів некоректні дані, він отримає сповіщення про це
Post-Condition	Потрапляння в профіль

Таблиця 2.1.2 – Варіант використання UC-2

Use case name	Реєстрація
Use case ID	UC-02
Goals	Реєстрація нового користувача в додатку
Actors	Гість
Trigger	Гість бажає створити обліковий запис щоб скористатися програмою
Pre-conditions	Натиснути кнопку «switch to register»
Flow of Events	Користувач відкриває додаток, натискає «switch to register». Користувач вводить свої дані, натискає кнопку і потрапляє в свій профіль
Extension	Якщо користувач не хоче реєструватися, він може повернутися до сторінки входу за допомогою кнопки «switch to login»
Post-Condition	Потрапляння в профіль

Таблиця 2.1.3 – Варіант використання UC-3

Use case name	Перегляд кількості оброблених зображень
Use case ID	UC-03
Goals	Демонстрація статистики по обробці зображень

Продовження таблиці 2.1.3

Actors	Користувач
Trigger	Користувач бажає знати кількість зображень які він обробив
Pre-conditions	Користувач авторизувався
Flow of Events	Користувач заходить в свій профіль. Користувач переглядає інформацію про себе.
Extension	-
Post-Condition	Користувач бачить лічильник з кількістю оброблених фото

Таблиця 2.1.4 – Варіант використання UC-4

Use case name	Перегляд інформації про бренди, що знайдені на зображенні
Use case ID	UC-04
Goals	Дізнатися які бренди знайдені на фото
Actors	Користувач
Trigger	Користувач вирішує дізнатися які бренди застосунок знайшов на фото
Pre-conditions	Користувач авторизувався
Flow of Events	Користувач заходить в свій профіль. Користувач обробляє фото і бачить назву бренду на рамці
Extension	-
Post-Condition	Користувач отримує інформацію про знайдений бренд

Таблиця 2.1.5 - Варіант використання UC-5

Use case name	Обробка фото
Use case ID	UC-05
Goals	Обробити фото
Actors	Користувач
Trigger	Користувач хоче обробити фотографію

Продовження таблиці 2.1.5

Pre-conditions	Авторизуватися в застосунку
Flow of Events	Користувач натискає кнопку обробки і вибирає фото яке хоче обробити. Фото обробляється
Extension	-
Post-Condition	Фото обробляється нейронною мережею

Таблиця 2.1.6 - Варіант використання UC-6

Use case name	Перегляд обробленого фото
Use case ID	UC-06
Goals	Побачити результат обробки
Actors	Користувач
Trigger	Користувач хоче переглянути результат обробки
Pre-conditions	Користувач має авторизуватися і почати обробку фото
Flow of Events	Фото відправлене в обробку. Користувач очікує результат.
Extension	-
Post-Condition	Користувач бачить у вікні програми оброблене фото

Таблиця 2.1.7 - Варіант використання UC-7

Use case name	Збереження фото
Use case ID	UC-07
Goals	Зберегти фото
Actors	Користувач
Trigger	Користувач хоче зберегти оброблене фото
Pre-conditions	Користувач має авторизуватися і почати обробку фото. Фото має бути обробленим
Flow of Events	Користувач натискає кнопку «Save». Користувач обирає директорію для збереження.
Extension	-

Продовження таблиці 2.1.7

Post-Condition	Фото збережено

Таблиця 2.1.8 - Варіант використання UC-8

Use case name	Вихід
Use case ID	UC-08
Goals	Вийти з облікового запису
Actors	Користувач
Trigger	Користувач хоче вийти з облікового запису
Pre-conditions	Користувач має авторизуватися
Flow of Events	Користувач натискає кнопку «Logout».
Extension	-
Post-Condition	Вихід здійснено

2.2 Аналіз системних вимог

Програмне забезпечення повинно функціонувати на IBM-сумісних персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core 2 Duo або аналоги від AMD;
– об‘єм ОЗП: 4 Гб;
– Windows 10 64-біт, Windows 8 64-біт.

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i3/i5/i7 або аналоги від AMD ;
– об‘єм ОЗП: 8 Гб;
– Windows 10 64-біт, Windows 8 64-біт.

2.3 Розроблення функціональних вимог

Програмне забезпечення розділене на три модулі. Кожен модуль має свій певний набір функцій. У таблиці 2.3.1 наведено загальну модель вимог, а в таблицях 2.3.2 – 2.3.12 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити на рисунку 2.3.1.

Таблиця 2.3.1 – Модель вимог у загальному вигляді

Номер	Опис	Код	Пріоритет	Ризик
1	Екранна форма авторизації	FR-1	2	Низький
2	Екранна форма профілю користувача	FR-2	1	Низький
3	Екранна форма реєстрації	FR-3	2	Низький
2.1	Відображення даних користувача	FR-4	4	Низький
2.2	Відображення лічильника кількості оброблених зображень	FR-5	3	Низький
1.1	Відображення повідомлення про неправильно введені дані на вході	FR-6	3	Низький
2.3	Вибір зображення для обробки	FR-7	1	Низький
2.4	Обробка та відображення обробленого зображення	FR-8	2	Високий
2.5	Вибір директорії та збереження обробленого зображення	FR-9	2	Низький
2.6	Можливість виходу з аккаунту користувача	FR-10		Низький
2.7	Відображення інформації про знайдені бренди	FR-11		Високий

Таблиця 2.3.2 – Функціональна вимога FR-1

Назва	Екранна форма авторизації
Опис	Система повинна при запуску виводити форму для авторизації

Таблиця 2.3.3 – Функціональна вимога FR-2

Назва	Екранна форма профілю користувача
Опис	На цій формі знаходиться основний функціонал проекту

Таблиця 2.3.4 – Функціональна вимога FR-3

Назва	Екранна форма реєстрації
Опис	Надає можливість створити обліковий запис

Таблиця 2.3.5 – Функціональна вимога FR-4

Назва	Відображення даних користувача
Опис	Система повинна відображати деталі про користувача (email, login)

Таблиця 2.3.6 – Функціональна вимога FR-5

Назва	Відображення лічильника кількості оброблених зображень
Опис	Система повинна відображати лічильник оброблених зображень

Таблиця 2.3.7 – Функціональна вимога FR-6

Назва	Відображення повідомлення про неправильно введені дані на вході

Таблиця 2.3.8 – Функціональна вимога FR-7

Назва	Вибір зображення для обробки
Опис	При натисканні кнопки для обробки зображення, система має відкрити файлову систему комп’ютера і дати можливість обрати фото для обробки

Таблиця 2.3.9 – Функціональна вимога FR-8

Назва	Обробка та відображення обробленого зображення
Опис	Система має знайти на зображенні логотип, накласти на нього розмиття і вивести результат на екран

Таблиця 2.3.10 – Функціональна вимога FR-9

Назва	Вибір директорії та збереження обробленого зображення

Таблиця 2.3.11 – Функціональна вимога FR-10

Назва	Можливість виходу з аккаунту користувача

Таблиця 2.3.12 – Функціональна вимога FR-11

Назва	Відображення інформації про підтримуванні бренди
Опис	Система має демонструвати рамку навколо знайденого логотипу і підписувати який саме бренд вона знайшла

	FR-1	FR-2	FR-3	FR-4	FR-5	FR-6	FR-7	FR-8	FR-9	FR-10	FR-11
UC-1	+					+					
UC-2			+								
UC-3		+		+	+						
UC-4		+						+			+
UC-5		+					+	+			
UC-6		+						+			
UC-7		+						+	+		
UC-8		+								+	

Рисунок 2.3.1 – Матриця трасування вимог

2.4 Розроблення нефункціональних вимог

У таблиці 2.4.1 наведено опис не функціональної вимоги до програмного забезпечення.

Таблиця 2.4.1

Назва	Можливість накладання ефекту розмиття на логотип має бути реалізована за допомогою ШІ

Висновки до розділу

Отже, після аналізу предметної області та після аналізу застосунків, що є потенційними конкурентами було сформовано функціональні вимоги, які роблять застосунок «Logo Detector» гарним додатковим інструментом на ринку застосунків для обробки фото. Адже більшість фоторедакторів не надають можливості автоматизувати пошук логотипів і їх заблюрення.

Також було проаналізовано та обрано допоміжні програмні застосунки та бібліотеки що є найбільш ефективними для розробки саме цього застосунку.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Для проектування самого застосунку було обрано патерн проектування MVC, що містить такі компоненти:

– M – компонент «Model», клас `image_detector` може вважатися моделлю у змісті архітектури MVC, оскільки він відповідає за виконання конкретної функціональності (детекція логотипів на фото);

– V – компонент «View», що відповідає за відображення інформації користувачу реалізований як клас `main`, в якому є екранні форми, і він виконує роль представлення (відображення даних) і взаємодії з користувачем. Він організовує та відображає інтерфейс користувача;

– C – компонент «Controller», це теж клас `main` який також містить функції авторизації, що вказує на те, що він також виконує роль контролера, керуючи логікою взаємодії між моделлю та виглядом.

Отже, архітектура включає в себе концепції MVC, де `image_detector` виступає в якості моделі, а `main` - як вигляд та контролер. Це може бути досить ефективним для розділення логіки додатку на окремі компоненти, що полегшує розширення та підтримку коду.

На рисунку 3.1.1 зображено загальний вигляд архітектури MVC [4]. Архітектура для застосунку «Logo Detector» представлена на демонстраційному плакаті 2.

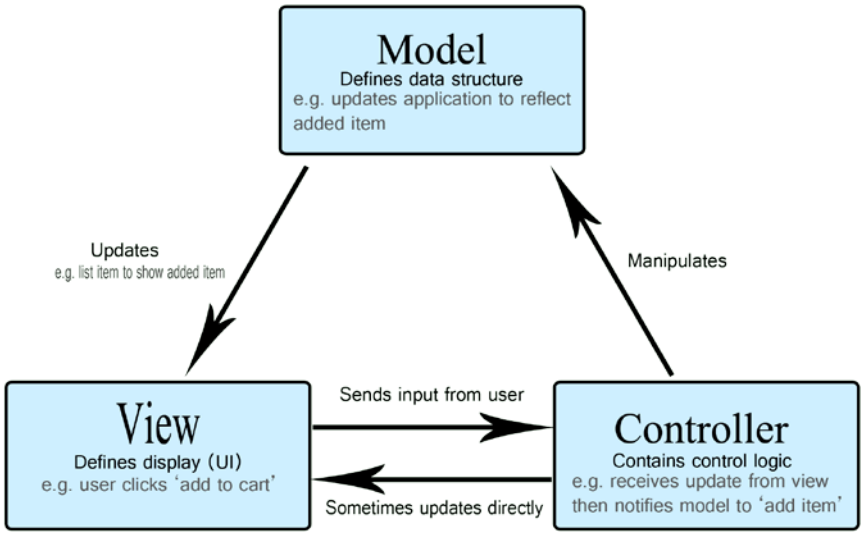


Рисунок 3.1.1 – Архітектура MVC

При запуску програми користувач взаємодіє з вікнами що реалізовані в класі main. Дані про його дії передаються в клас image_detector або напряму впливають на View. Також компонент Model відправляє дані в треновану модель, яка повертає йому результат обробки. Діаграма архітектури «Pre-trained model» являє собою архітектуру YOLO v8 [6]. Вона відображена на рисунку 3.1.2 і у кращій якості на демонстраційному плакаті 2.

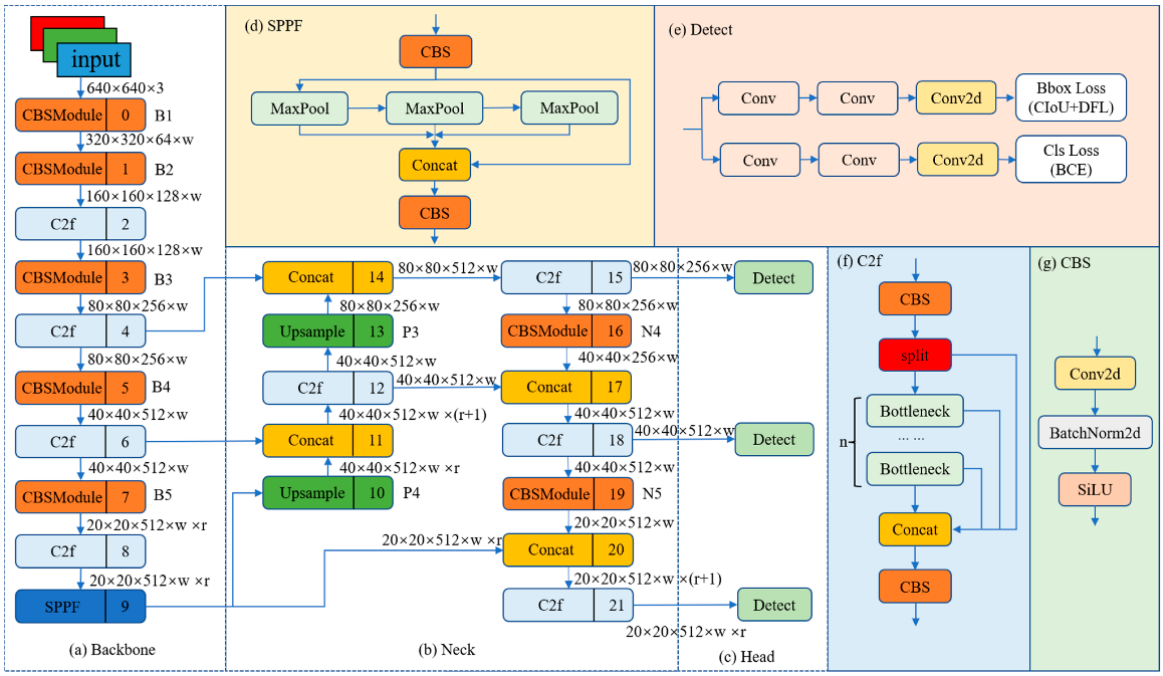


Рисунок 3.1.2 – Архітектура «Pre-trained model»

3.2 Обґрунтування вибору засобів розробки

Мовою програмування була обрана Python. Ця мова є лідером у вирішенні задач з машинного навчання.

Для задачі детекції був використаний алгоритм машинного навчання на основі архітектури YOLOv8 s. Алгоритм був обраний тому, що використовує сверточну нейронну мережу (CNN) для розпізнавання об'єктів на зображеннях. Основна ідея полягає в тому, щоб ввести на вхід всю картинку, яка проходить через сверточну нейронну мережу лише один раз, на відміну від інших аналогічних алгоритмів. Це значно знижувало вимоги до потужностей при розробці.

3.3 Конструювання програмного забезпечення

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.3.1:

Таблиця 3.3.1 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	PyCharm	Головне середовище розробки програмного забезпечення курсової роботи.
2	Ultralytics	Бібліотека для роботи з моделями YOLO

Бібліотека Ultralytics була обрана для цього проекту через її високу ефективність та простоту використання у задачах комп'ютерного зору. Ultralytics спеціалізується на розробці та впровадженні моделей для обробки зображень, зокрема, моделей для виявлення об'єктів на основі архітектури

YOLO (You Only Look Once). Ця архітектура відома своєю швидкістю і точністю, що робить її ідеальною для застосувань у реальному часі.

У проекті обробки фотографій, який включає в себе функції входу користувачів, реєстрації та обробки зображень, Ultralytics використовується для виявлення логотипів на зображеннях. Завдяки цій бібліотеці, ми можемо виконувати об'єктне розпізнавання на завантажених фотографіях, а також застосовувати розмиття до виявлених об'єктів. Це дозволяє автоматично обробляти зображення з високою точністю і швидкістю, що є критично важливим для користувацького досвіду.

Ще однією причиною вибору Ultralytics є активна підтримка та оновлення з боку спільноти та розробників. Це гарантує, що бібліотека завжди знаходиться на передовій сучасних технологій у галузі комп'ютерного зору. Документація Ultralytics також дуже детальна і доступна, що полегшує процес навчання та впровадження для нових користувачів. Усі ці фактори роблять Ultralytics оптимальним вибором для реалізації задач з виявлення та обробки об'єктів у цьому проекті.

Тексти програмного коду наведені в окремому документі «Текст програми».

3.4 Аналіз безпеки даних

Локальне сховище даних було реалізовано за допомогою json файлу. Таке рішення обумовлене тим, що до програми не висувається ніяких вимог по безпеці даних і програма не зберігає ніяких конфіденційних даних.

Висновки до розділу

Отже, в розділі було змодельовано архітектуру застосунку на основі патерну MVC, описано необхідні допоміжні бібліотеки для роботи з моделями, описано архітектуру обраної моделі для навчання та її переваги.

Було обрано та описано утиліту для розробки: PyCharm як головна IDE.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження даних;

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

Метриками для оцінки якості ПЗ обрано наступні:

– Точність детекції (за внутрішніми метриками бібліотеки ultralytics);

– Кількість підтримуваних брендів.

Програмне забезпечення застосунку «Logo Detector» було проаналізовано за допомогою вбудованих в бібліотеку ultralytics тестів моделі та мануальних тестів. Якість моделі проаналізовано за кількома метриками (Рис. 4.1.1):

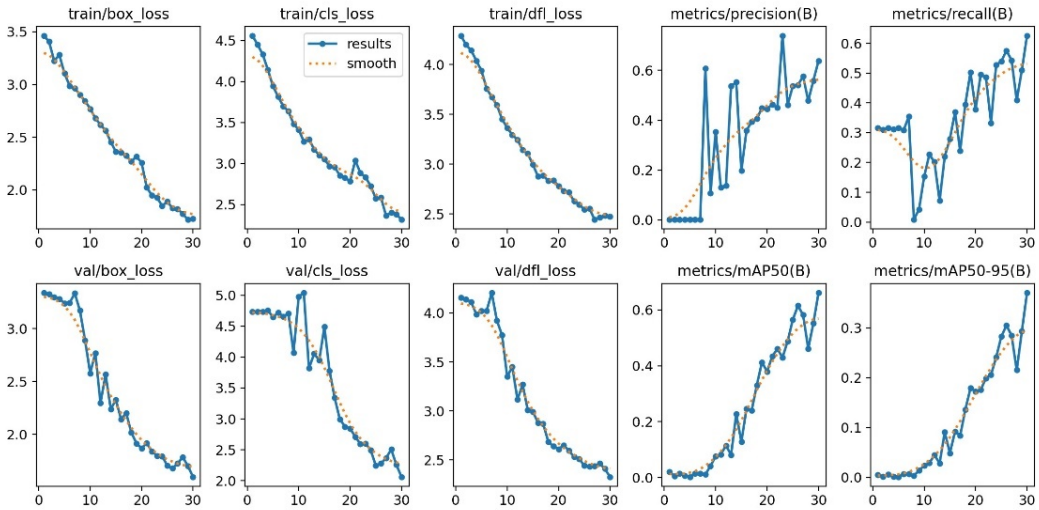


Рисунок 4.1.1 – Результати аналізу якості.

Також можна побачити результати аналізу якості детекції зображень на матриці плутанини (Рис. 4.1.2). В кращій якості ці графіки можна побачити на демонстраційних плакатах 3-4.

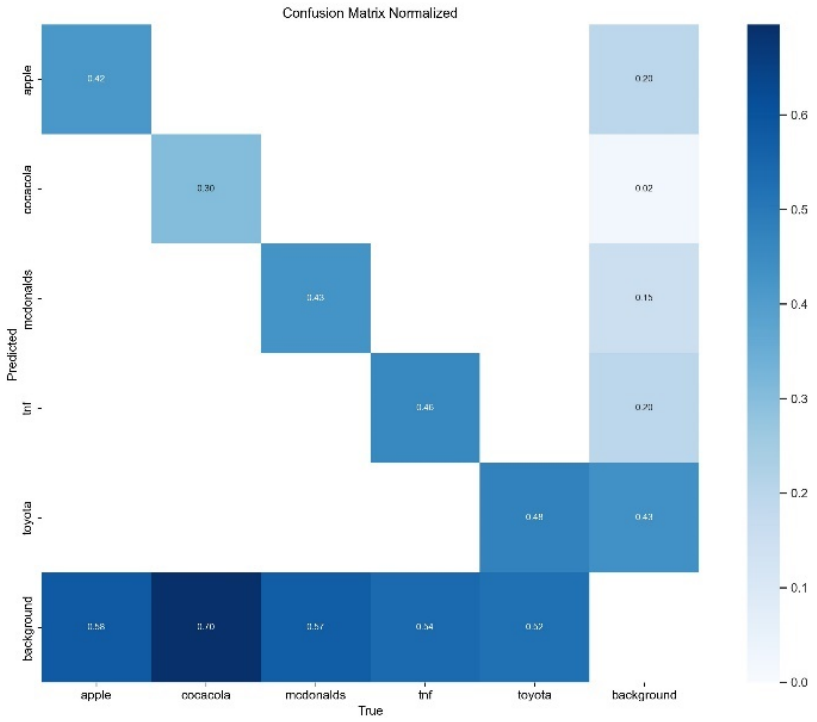


Рисунок 4.1.2 – матриця плутанини.

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.2.1 –4.2.8:

Таблиця 4.2.1 – Тест 1.1

Тест	Авторизація
Модуль	Вікно авторизації
Номер тесту	1.1
Початковий стан системи	Користувач відкрив щойно встановлений застосунок.

Продовження таблиці 4.2.1

Опис проведення тесту	При першому запуску застосунку має з’явитися запит до користувача про введення даних.
Очікуваний результат	Користувач вводить дані і потрапляє в додаток.
Фактичний результат	Користувач вводить дані і потрапляє в додаток.

Таблиця 4.2.2 – Тест 1.2

Тест	Реєстрація
Модуль	Вікно реєстрації
Номер тесту	1.2
Початковий стан системи	Користувач натиснув кнопку «Switch to register».
Опис проведення тесту	При переході на вікно реєстрації користувач має заповнити дані про себе.
Очікуваний результат	Користувач успішно створює обліковий запис.
Фактичний результат	Користувач успішно створює обліковий запис.

Таблиця 4.2.3 – Тест 2.1

Тест	Перегляд даних користувача
Модуль	Вікно профілю
Номер тесту	2.1
Початковий стан системи	Користувач авторизувався

Продовження таблиці 4.2.3

Опис проведення тесту	Після авторизації користувач має побачити дані про себе.
Очікуваний результат	Дані відображаються правильно.
Фактичний результат	Дані відображаються правильно.

Таблиця 4.2.4 – Тест 2.2

Тест	Перевірка лічильника обробок
Модуль	Вікно профілю
Номер тесту	2.2
Початковий стан системи	Користувач авторизувався
Опис проведення тесту	Користувач переглядає лічильник, обробляє фото і фіксує чи змінилося число.
Очікуваний результат	Лічильник враховує обробку фото.
Фактичний результат	Лічильник врахував обробку фото.

Таблиця 4.2.5 – Тест 3.1

Тест	Обробка фотографії
Модуль	Вікно профілю
Номер тесту	3.1
Початковий стан системи	Користувач авторизувався

Продовження таблиці 4.2.5

Опис проведення тесту	Користувач натискає кнопку обробки, обирає файл, файл обробляється і виводиться результат на екран.
Очікуваний результат	На екрані фото з виявленим логотипом і ефектом розмиття на ньому.
Фактичний результат	На екрані фото з виявленим логотипом і ефектом розмиття на ньому.

Таблиця 4.2.6 – Тест 3.2

Тест	Збереження обробленої фотографії
Модуль	Вікно профілю
Номер тесту	3.2
Початковий стан системи	Користувач авторизувався і обробив фото
Опис проведення тесту	Користувач тисне кнопку «save»
Очікуваний результат	Відкривається вікно вибору директорії для збереження. Після збереження, фото містить розмиті логотипи
Фактичний результат	Відкривається вікно вибору директорії для збереження. Після збереження, фото містить розмиті логотипи

Таблиця 4.2.7 – Тест 3.3

Тест	Перегляд знайденого бренду
Модуль	Вікно профілю
Номер тесту	3.3
Початковий стан системи	Користувач авторизувався і обробив фото

Продовження таблиці 4.2.7

Опис проведення тесту	Користувач переглядає оброблене фото
Очікуваний результат	Знайдений логотип підсвічується рамкою-підписом з назвою бренду
Фактичний результат	Знайдений логотип підсвічується рамкою-підписом з назвою бренду

Таблиця 4.2.8 – Тест 3.4

Тест	Вихід з облікового запису
Модуль	Вікно профілю
Номер тесту	3.2
Початковий стан системи	Користувач авторизувався
Опис проведення тесту	Користувач тисне кнопку виходу
Очікуваний результат	Відкривається вікно авторизації
Фактичний результат	Відкривається вікно авторизації

4.3 Опис контрольного прикладу

Першим кроком слід авторизуватися в додатку. Для цього створимо новий профіль з ім’ям Oleksiy і вигаданою електронною поштою (рис. 4.3.1)

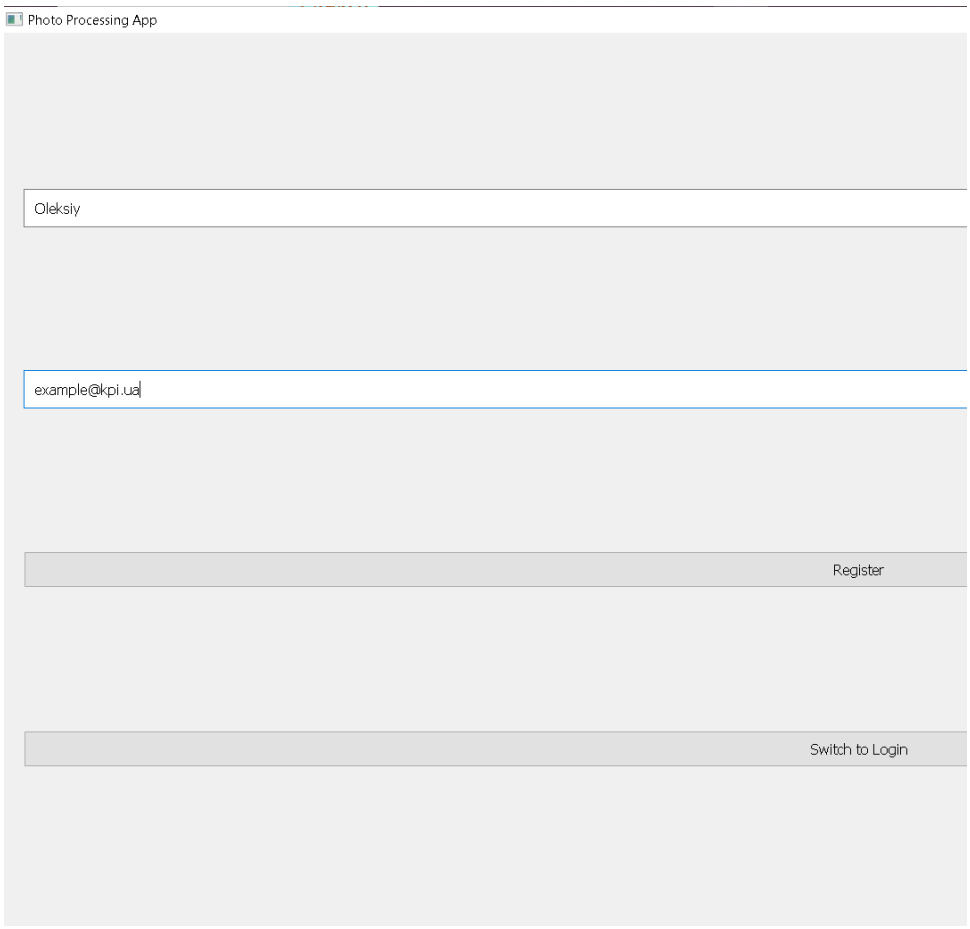


Рисунок 4.3.1 – Реєстрація нового користувача.

Наступним кроком ми потрапляємо у профіль користувача і можемо переглянути лічильник і дані користувача (рис. 4.3.2)

Рисунок 4.3.2 – Профіль користувача.

Тепер натискаємо кнопку для обробки фото і обираємо те, яке хочемо обробити. Для прикладу візьмемо фото ноутбуку з логотипом компанії “Apple”. На рисунках 4.3.3 – 4.3.4 можемо побачити процес:

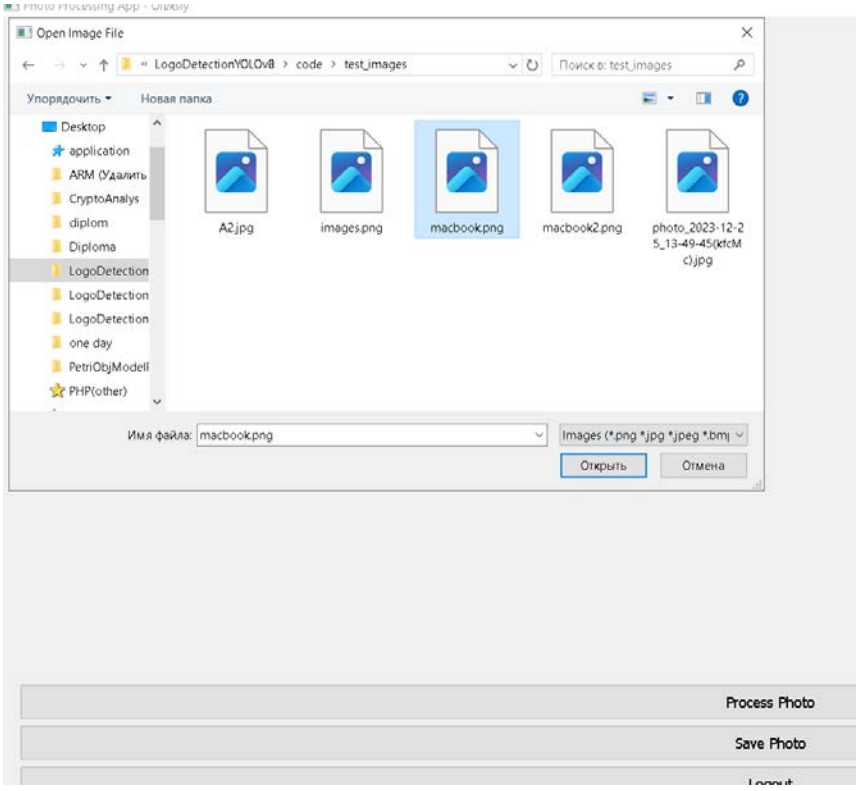


Рисунок 4.3.3 – Вибір фото.

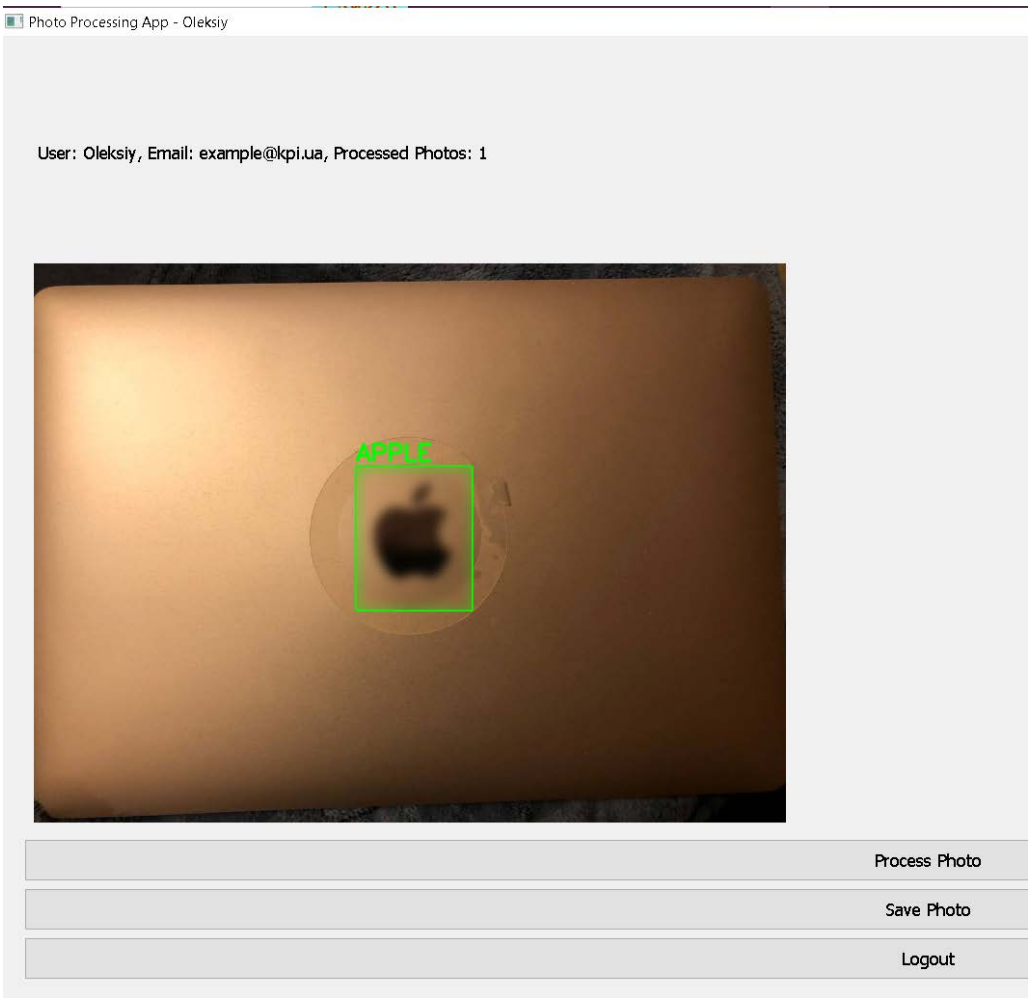


Рисунок 4.3.4 – Результат обробки.

В результаті можемо побачити виконання одразу кількох функціональних вимог, а саме збільшення значення лічильника оброблених фото, відображення інформації про знайдений логотип і накладання ефекту розмиття.

Тепер збережемо це фото. Для цього натискаємо кнопку “Save Photo” і обираємо директорію. Після цього з’явиться повідомлення про успішне збереження фотографії. Результати можна побачити на рисунках 4.3.5-4.3.7:

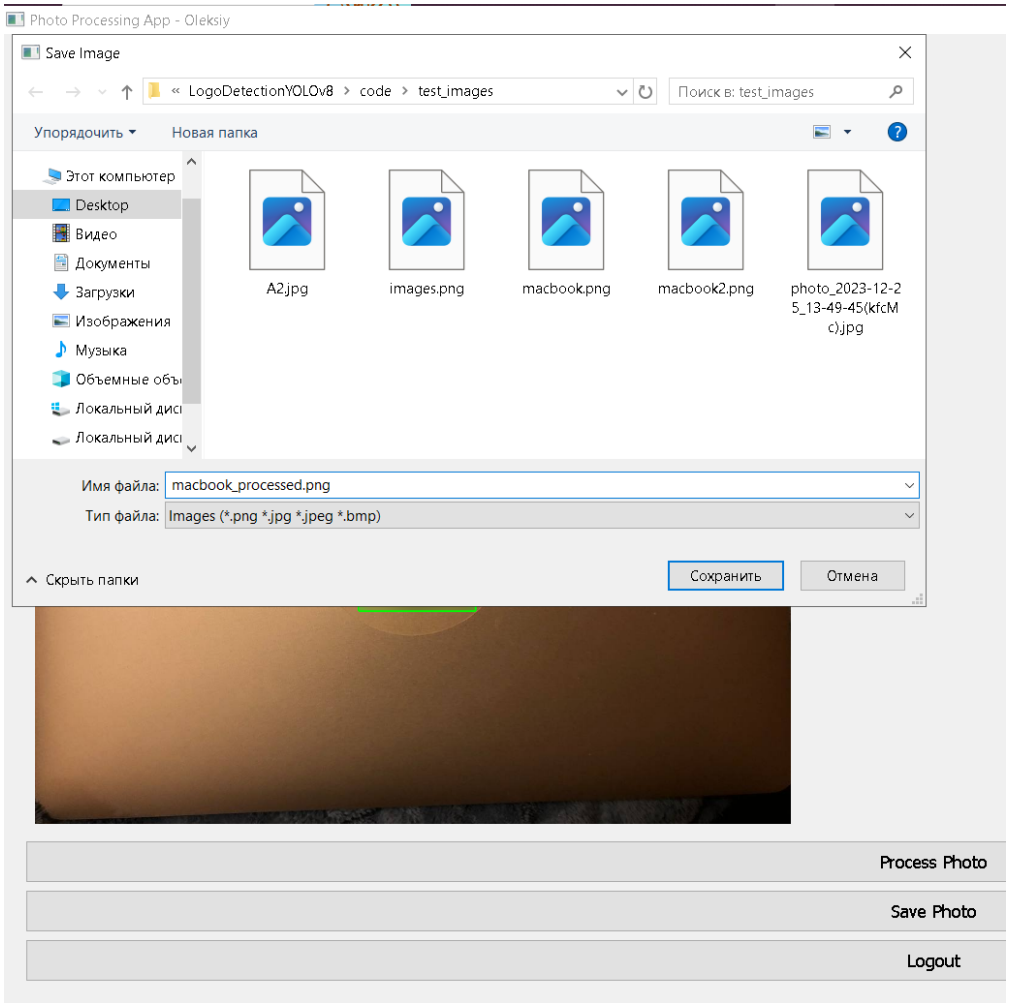


Рисунок 4.3.5 – Вибір директорії.

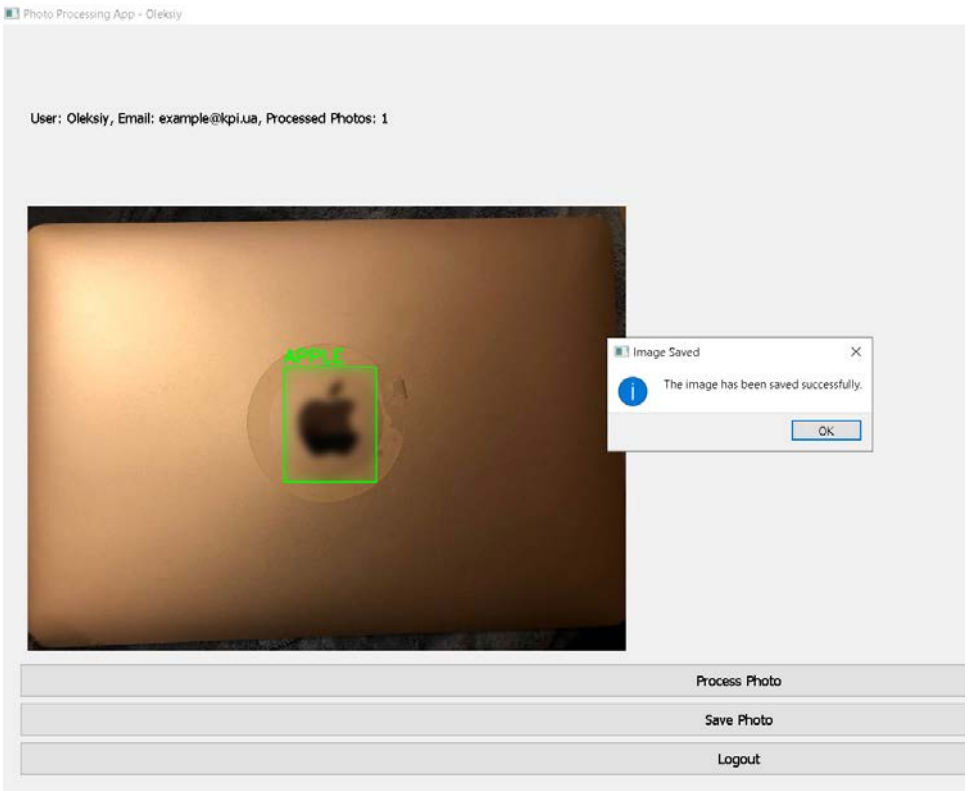


Рисунок 4.3.6 – Успішне збереження.

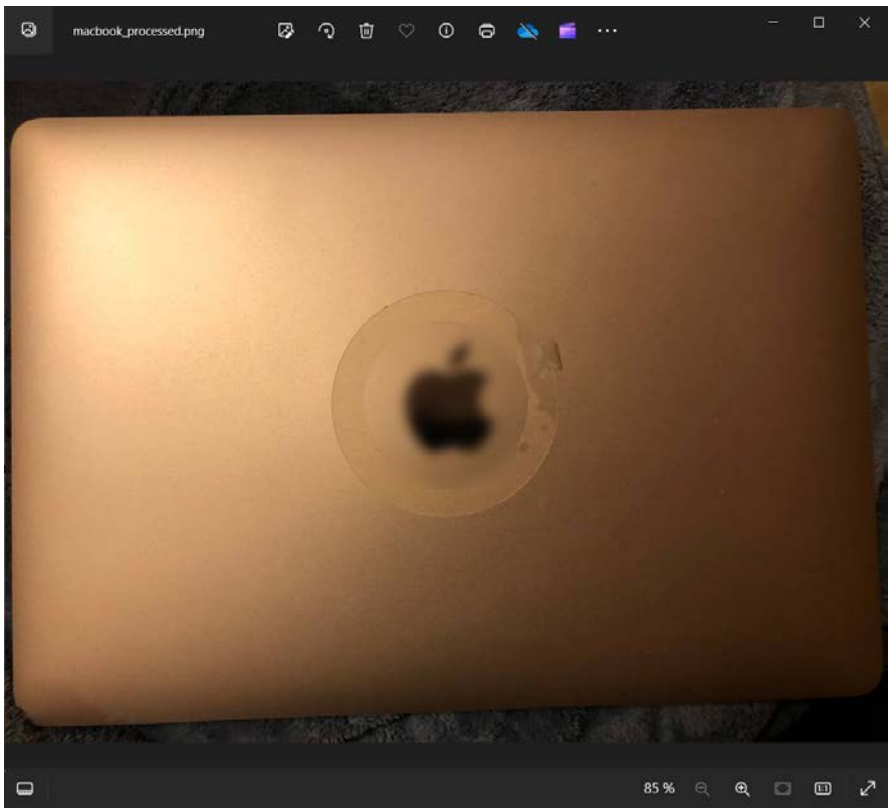


Рисунок 4.3.7 – Результат.

Висновки до розділу

Отже, у розділі було проаналізовано якість програмного забезпечення та не знайдено проблем чи помилок, що могли б критично вплинути на роботу застосунку.

Також було розроблено список тестів мануального тестування застосунку для того, щоб переконатися у відповідності програмного забезпечення.

В ході опису контрольного прикладу, було також продемонстровано роботу основного функціоналу додатку, продемонстровано правильне виконання функціональних вимог.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Моя програма, спрямована на детекцію логотипів на фотографіях, розроблена з метою надання відкритого доступу користувачам на різних платформах. Я прагну зробити її доступною для широкого кола користувачів, надаючи можливість використовувати це програмне забезпечення для обробки фотографій абсолютно безкоштовно.

Розгортання програми на різних платформах, зокрема для операційної системи Windows, відкриває двері для більшого кола користувачів. Я хочу, щоб моя технологія детекції логотипів була доступною для різних користувачів, незалежно від їх технічного рівня.

Забезпечуючи відкритий доступ до моєї програми на платформі Windows, я надаю можливість користувачам використовувати цей інструмент для обробки та аналізу фотографій з абсолютною вільністю. Це відкриває широкі перспективи для використання мого рішення в різних сферах та галузях.

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі.

Моя програма для детекції логотипів на фотографіях розроблена з урахуванням технічних вимог та потреб користувачів. Планується надавати систематичні технічні оновлення, спрямовані на виправлення поточних недоліків та покращення алгоритмів детекції.

У майбутньому планується розширення функціональності, включаючи підтримку нових брендів та зростання кількості об'єктів, які можуть бути розпізнані. Можливість обробляти декілька фото за раз. Також буде внесена можливість детекції логотипів на відео, що розширить область застосування програми. В планах додати можливість надсилати фото по електронній пошті.

Забезпечення стабільності та продуктивності є пріоритетом, і ми активно вивчатимемо нові технології та методи для вдосконалення якості детекції. Користувачі можуть очікувати систематичних оновлень з метою оптимізації та розширення можливостей програмного забезпечення для більш ефективного використання у різних сценаріях.

Висновки до розділу

В даному розділі було наведено плани по розгортанню та супроводу даного програмного забезпечення. Вказано платформи на яких відбуватиметься розгортання.

Було описано заплановані доробки в наступних версіях застосунку і зазначено модель монетизації, а саме повний безкоштовний доступ до усього функціоналу.

ВИСНОВКИ

В результаті виконання дипломного проєкту було досягнуто автоматизації знаходження та розмиття логотипів 5 різних брендів на фото шляхом створення застосунку для виявлення логотипів брендів на основі штучного інтелекту.

В якості середовища розробки обрано PyCharm, адже він надає зручний інтерфейс для роботи з Python і його бібліотеками.

Після реалізації застосунку він був протестований на різних фотографіях, з різними логотипами щоб переконатися, що додаток правильно знаходить і розпізнає різні бренди.

Програма має простий та інтуїтивний інтерфейс, що робить її доступною для широкого кола користувачів. Алгоритми детекції логотипів дозволяють ефективно визначати об'єкти із зображень, що важливо для багатьох застосувань, таких як редагування фотографій, відстеження брендів та аналіз зображень.

За допомогою технологій машинного навчання та обробки зображень, програма поєднує в собі точність та швидкість у виявленні логотипів. Проте, завданням майбутніх вдосконалень є покращення алгоритмів, розширення підтримуваних брендів та введення можливості детекції на відео.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Cvat. *CVAT*. URL: https://www.cvat.ai/ (дата звернення: 11.06.2024).
2) Draw.io. *draw.io*.URL: https://www.drawio.com/ (дата звернення: 11.06.2024).
3) GitHub - ultralytics/ultralytics: NEW - YOLOv8 🚀 in PyTorch > ONNX >OpenVINO>CoreML>TFLite. *GitHub*.URL: https://github.com/ultralytics/ultralytics (дата звернення: 11.06.2024).
4) MVC - MDN web docs glossary: definitions of web-related terms | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Glossary/MVC (дата звернення: 11.06.2024).
5) Raunak agrawal. Text to knowledge – NLP in social media. *Medium*. URL: https://raunakagrawal11.medium.com/text-to-knowledge-nlp-in-social-media-d552cf26dbb4 (дата звернення: 11.06.2024).
6) UAV-YOLOv8: A small-object-detection model based on improved yolov8 for UAV aerial photography scenarios. *MDPI*. URL: https://www.mdpi.com/1424-8220/23/16/7190 (дата звернення: 11.06.2024).

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ЗАСТОСУНОК ДЛЯ ВИЯВЛЕННЯ ЛОГОТИПІВ БРЕНДІВ НА ФОТО НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

Текст програми

КПІ.ІТ-0125.045480.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Валерій НОВІНСЬКИЙ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Антон ТЕРЕШПОЛЬСЬКИЙ

Київ – 2024

Посилання на репозиторій з повним текстом програмного коду
https://github.com/smashok/IT-0125.git

Файл main.py

```
import sys
from PyQt5.QtWidgets import QApplication, QMainWindow, QLabel,
QVBoxLayout, QPushButton, QFileDialog, QWidget, QLineEdit, QMessageBox,
QStackedWidget
from PyQt5.QtGui import QPixmap, QImage
from PyQt5.QtCore import Qt
import numpy as np
import json
from image_detector import perform_object_detection, apply_blur
import cv2

class User:
    def __init__(self, name, email):
        self.name = name
        self.email = email
        self.processed_photos = 0

class PhotoProcessingApp(QMainWindow):
    def __init__(self):
        super(PhotoProcessingApp, self).__init__()

        self.users = []
        self.processed_image = None

        self.initUI()

    def initUI(self):
        self.setWindowTitle('Photo Processing App')
        self.setWindowState(Qt.WindowMaximized)

        self.central_widget = QWidget(self)
        self.setCentralWidget(self.central_widget)

        self.stacked_widget = QStackedWidget(self.central_widget)


        self.login_widget = self.create_login_widget()
        self.register_widget = self.create_register_widget()
        self.main_widget = self.create_main_widget()
```

```
        self.stacked_widget.addWidget(self.login_widget)
        self.stacked_widget.addWidget(self.register_widget)
        self.stacked_widget.addWidget(self.main_widget)

        self.central_layout = QVBoxLayout()
        self.central_layout.addWidget(self.stacked_widget)
        self.central_widget.setLayout(self.central_layout)

        self.user = None


        self.load_users()


        self.stacked_widget.setCurrentWidget(self.login_widget)

    def create_login_widget(self):
        login_widget = QWidget(self)

        login_layout = QVBoxLayout()

        self.login_name_edit = QLineEdit(self)
        self.login_name_edit.setPlaceholderText('Enter your name')
        login_layout.addWidget(self.login_name_edit)

        self.login_email_edit = QLineEdit(self)
        self.login_email_edit.setPlaceholderText('Enter your email')
        login_layout.addWidget(self.login_email_edit)

        login_button = QPushButton('Login', self)
        login_button.clicked.connect(self.login_user)
        login_layout.addWidget(login_button)

        switch_to_register_button = QPushButton('Switch to Register', self)
        switch_to_register_button.clicked.connect(lambda:
self.stacked_widget.setCurrentWidget(self.register_widget))
        login_layout.addWidget(switch_to_register_button)

        login_widget.setLayout(login_layout)

        login_widget.setStyleSheet("""
            QLineEdit {
                font-size: 16px;
                padding: 10px;
```

```
            }
            QPushButton {
                font-size: 16px;
                padding: 10px;
            }
        """)

        return login_widget

    def create_register_widget(self):
        register_widget = QWidget(self)

        register_layout = QVBoxLayout()

        self.register_name_edit = QLineEdit(self)
        self.register_name_edit.setPlaceholderText('Enter your name')
        register_layout.addWidget(self.register_name_edit)

        self.register_email_edit = QLineEdit(self)
        self.register_email_edit.setPlaceholderText('Enter your email')
        register_layout.addWidget(self.register_email_edit)

        register_button = QPushButton('Register', self)
        register_button.clicked.connect(self.register_user)
        register_layout.addWidget(register_button)

        switch_to_login_button = QPushButton('Switch to Login', self)
        switch_to_login_button.clicked.connect(lambda:
self.stacked_widget.setCurrentWidget(self.login_widget))
        register_layout.addWidget(switch_to_login_button)

        register_widget.setLayout(register_layout)

        register_widget.setStyleSheet("""
            QLineEdit {
                font-size: 16px;
                padding: 10px;
            }
            QPushButton {
                font-size: 16px;
                padding: 10px;
            }
        """)

        return register_widget
```

```
def create_main_widget(self):
    main_widget = QWidget(self)

    main_layout = QVBoxLayout()

    self.user_info_label = QLabel(self)
    main_layout.addWidget(self.user_info_label)

    self.label = QLabel(self)
    main_layout.addWidget(self.label)

    process_button = QPushButton('Process Photo', self)
    process_button.clicked.connect(self.process_photo)
    main_layout.addWidget(process_button)

    save_button = QPushButton('Save Photo', self)
    save_button.clicked.connect(self.save_photo)
    main_layout.addWidget(save_button)

    logout_button = QPushButton('Logout', self)
    logout_button.clicked.connect(self.logout_user)
    main_layout.addWidget(logout_button)

    main_widget.setLayout(main_layout)

    main_widget.setStyleSheet("""
        QLabel {
            font-size: 18px;
            padding: 10px;
        }
        QPushButton {
            font-size: 18px;
            padding: 10px;
        }
    """)

    return main_widget

def show_register_window(self):
    self.stacked_widget.setCurrentWidget(self.register_widget)

def show_login_window(self):
    self.stacked_widget.setCurrentWidget(self.login_widget)
```

```
    def show_main_window(self):
        self.stacked_widget.setCurrentWidget(self.main_widget)
        user_info = f'User: {self.user.name}, Email: {self.user.email}, Processed
Photos: {self.user.processed_photos}'
        self.user_info_label.setText(user_info)

    def load_users(self):
        try:
            with open('users.json', 'r') as file:
                data = json.load(file)
                for user_data in data:
                    user = User(user_data['name'], user_data['email'])
                    user.processed_photos = user_data.get('processed_photos', 0)
                    self.users.append(user)
        except FileNotFoundError:
            pass

    def save_users(self):
        with open('users.json', 'w') as file:
            data = [{'name': user.name, 'email': user.email, 'processed_photos':
user.processed_photos} for user in self.users]
            json.dump(data, file)

    def login_user(self):
        name = self.login_name_edit.text().strip()
        email = self.login_email_edit.text().strip()

        if not name or not email:
            QMessageBox.warning(self, 'Login Error', 'Please enter your name and
email.')
            return

        for user in self.users:
            if user.name == name and user.email == email:
                self.user = user
                self.setWindowTitle(f'Photo Processing App - {self.user.name}')
                self.show_main_window()
                return

        QMessageBox.warning(self, 'Login Error', 'User not found. Please register.')

    def register_user(self):
        name = self.register_name_edit.text().strip()
        email = self.register_email_edit.text().strip()
```

```
        if name and email:
            self.user = User(name, email)
            self.setWindowTitle(f'Photo Processing App - {self.user.name}')
            self.users.append(self.user)
            self.save_users()
            self.show_main_window()
        else:
            QMessageBox.warning(self, 'Registration Error', 'Please enter your name
and email.')

    def process_photo(self):
        if not self.user:
            QMessageBox.warning(self, 'User Not Logged In', 'Please log in or register
before processing photos.')
            return


        image_path, _ = QFileDialog.getOpenFileName(self, 'Open Image File', '',
'Images (*.png *.jpg *.jpeg *.bmp)')

        if image_path:
            self.last_image_path = image_path
            result, result_without_boxes = perform_object_detection(image_path,
return_result_without_boxes=True)
            self.processed_image = result_without_boxes  # Сохраняем обработанное
изображение без рамок
            self.show_result(result)



            self.user.processed_photos += 1
            self.save_users()



            self.show_main_window()

    def save_photo(self):
        if self.processed_image is None:
            QMessageBox.warning(self, 'No Image', 'There is no processed image to
save.')
            return



        save_path, _ = QFileDialog.getSaveFileName(self, 'Save Image', '', 'Images
(*.png *.jpg *.jpeg *.bmp)')
        if save_path:
```

```
            cv2.imwrite(save_path, self.processed_image)
            QMessageBox.information(self, 'Image Saved', 'The image has been saved
successfully.')

    def show_result(self, result):

        fixed_width = 800
        fixed_height = 600


        result = cv2.resize(result, (fixed_width, fixed_height))

        height, width, channel = result.shape
        bytes_per_line = 3 * width
        qImg = QImage(result.data, width, height, bytes_per_line,
QImage.Format_RGB888).rgbSwapped()
        pixmap = QPixmap.fromImage(qImg)


        self.label.setPixmap(pixmap)
        self.label.setFixedSize(fixed_width, fixed_height)  # Устанавливаем
фиксированный размер для QLabel
        self.label.setAlignment(Qt.AlignCenter)

    def logout_user(self):
        self.user = None
        self.setWindowTitle('Photo Processing App')
        self.show_login_window()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = PhotoProcessingApp()
    window.show()
    sys.exit(app.exec_())
```

Файл image_detector.py

```
import os

from ultralytics import YOLO

import cv2

import numpy as np
```

```
def apply_blur(image, x1, y1, x2, y2, blur_factor=55):

    roi = image[int(y1):int(y2), int(x1):int(x2)]

    blurred_roi = cv2.GaussianBlur(roi, (blur_factor, blur_factor), 0)

    image[int(y1):int(y2), int(x1):int(x2)] = blurred_roi
    return image

def perform_object_detection(image_path,
model_path='C:/Users/Anton/Desktop/LogoDetectionYOLOv8/code/runs/detect/tr
ain2/weights/last.pt', threshold=0.1, return_result_without_boxes=False):

    model = YOLO(model_path)



    frame = cv2.imread(image_path)
    H, W, _ = frame.shape
    original_frame = frame.copy()



    results = model(frame)[0]

    for result in results.boxes.data.tolist():
        x1, y1, x2, y2, score, class_id = result

        if score > threshold:

            cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 4)
```

```
            cv2.putText(frame, results.names[int(class_id)].upper(), (int(x1), int(y1 -
10)),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 255, 0), 3,
cv2.LINE_AA)

            frame = apply_blur(frame, x1, y1, x2, y2)
            original_frame = apply_blur(original_frame, x1, y1, x2, y2)

    if return_result_without_boxes:
        return frame, original_frame
    else:
        return frame
```

Файл config.yaml

```
path: C:/Users/Anton/Desktop/LogoDetectionYOLOv8/code/data
train: test_images/train
val: test_images/train

names:
  0: apple
  1: cocacola
  2: mcdonalds
  3: tnf
  4: toyota

augment:
    enable: True
    conf: 0.5
    degrees: 10
    translate: 0.1
    scale: 0.2
    shear: 0.1
    perspective: 0.1
    flipud: True
    fliplr: True
    mosaic: 0.2
    mixup: 0.2
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ЗАСТОСУНОК ДЛЯ ВИЯВЛЕННЯ ЛОГОТИПІВ БРЕНДІВ НА ФОТО НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

Програма та методика тестування

КПІ.ІТ-0125. 045480.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Валерій НОВІНСЬКИЙ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Антон ТЕРЕШПОЛЬСЬКИЙ

Київ – 2024

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є застосунок «Logo Detector» для ОС Windows з версією 8 та 10.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження даних;

– знаходження проблем, помилок і недоліків з метою їх усунення;

– перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

– статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;

– функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;

– мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення;

– внутрішнє автоматичне тестування моделі.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- мануальне тестування на відповідність функціональним вимогам;
- тестування інтерфейсу користувача;
- тестування зручності використання;
- тестування правильності і точності роботи.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ЗАСТОСУНОК ДЛЯ ВИЯВЛЕННЯ ЛОГОТИПІВ БРЕНДІВ НА ФОТО НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

Керівництво користувача

КПІ.ІТ-0125.045480.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Валерій НОВІНСЬКИЙ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____Антон ТЕРЕШПОЛЬСЬКИЙ

Київ – 2024

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

«LogoDetector» - це застосунок, що допомагає фотографам та людям які створюють медіа контент, уникати проблем з авторським правом шляхом автоматизованого виявлення логотипів брендів на фотографіях та подальшої обробки цих фото. Обробка фото передбачає накладання ефекту розмиття на ділянку що містить логотип певного бренду. Це дозволить використовувати фото в комерційних цілях без ризику зіштовхнутись з правовими наслідками.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

– наявність на пристрої Python 3.8 або більш нової версії;

– наявність зображень для обробки;

– для встановлення додатку на пристрої повинно бути не менше 10 МБ вільної пам’яті.

2.2 Завантаження застосунку

Застосунок можна завантажити архівом з джерел на які він буде викладений.

2.3 Перевірка коректної роботи

Після запуску застосунку має з’явитись вікно входу в обліковий запис, якщо такий існує. Якщо вікно не з’явилось, значить вимоги запуску було порушено і слід перевстановити застосунок.

3 ВИКОНАННЯ ПРОГРАМИ

Після запуску програми, користувач потрапляє на сторінку входу до облікового запису. На ній йому слід ввести свій унікальний логін і електронну пошту.

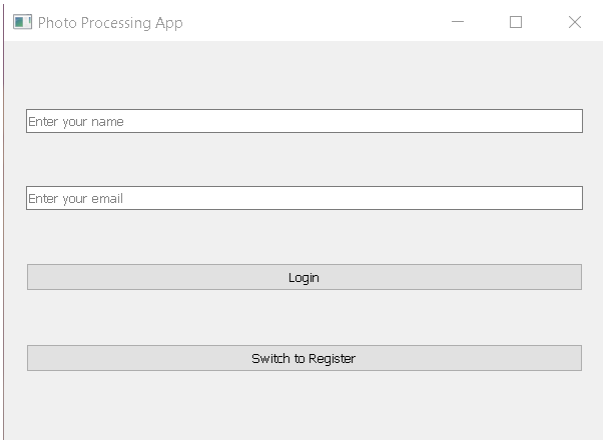


Рисунок 3.1 – сторінка входу

Якщо облікового запису немає, слід натиснути кнопку реєстрації і створити новий. Після цього користувач отримує доступ до основного функціоналу.

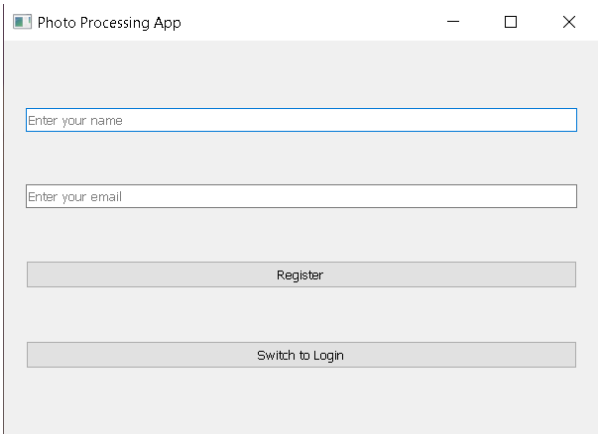


Рисунок 3.2 – сторінка створення облікового запису

На вікні профілю користувач може переглянути свої дані, а також лічильник кількості оброблених фотографій. Також він може виконати обробку зображення і побачити результат, який містить в собі інформацію про те, який бренд було знайдено.

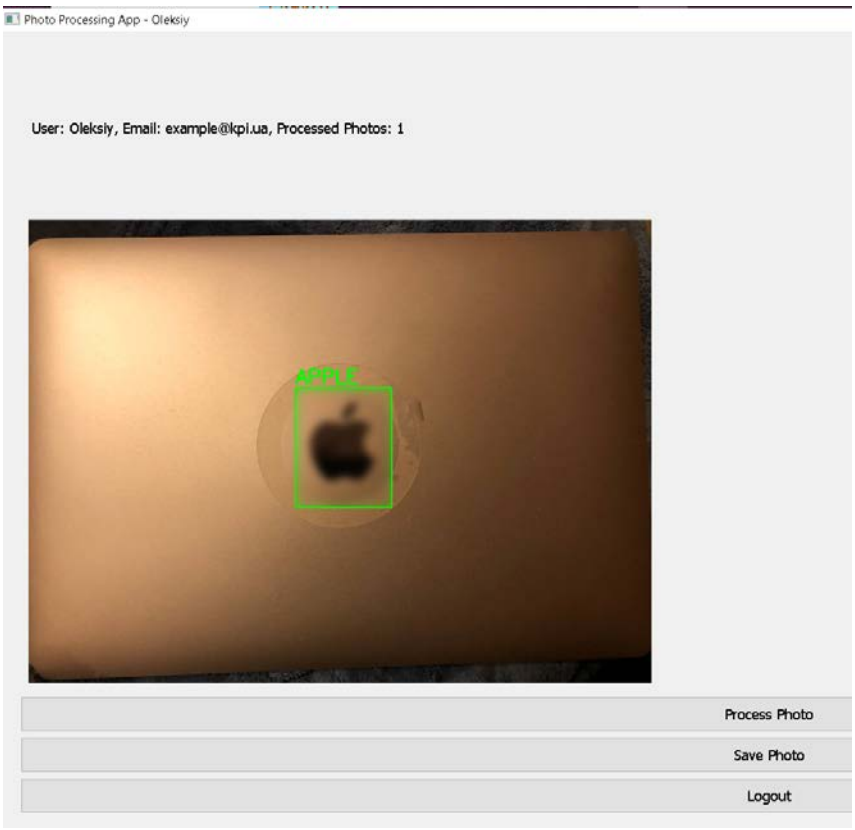


Рисунок 3.3 – сторінка профілю з обробленим фото

Після обробки користувач має змогу обрати директорію і зберегти оброблене фото.

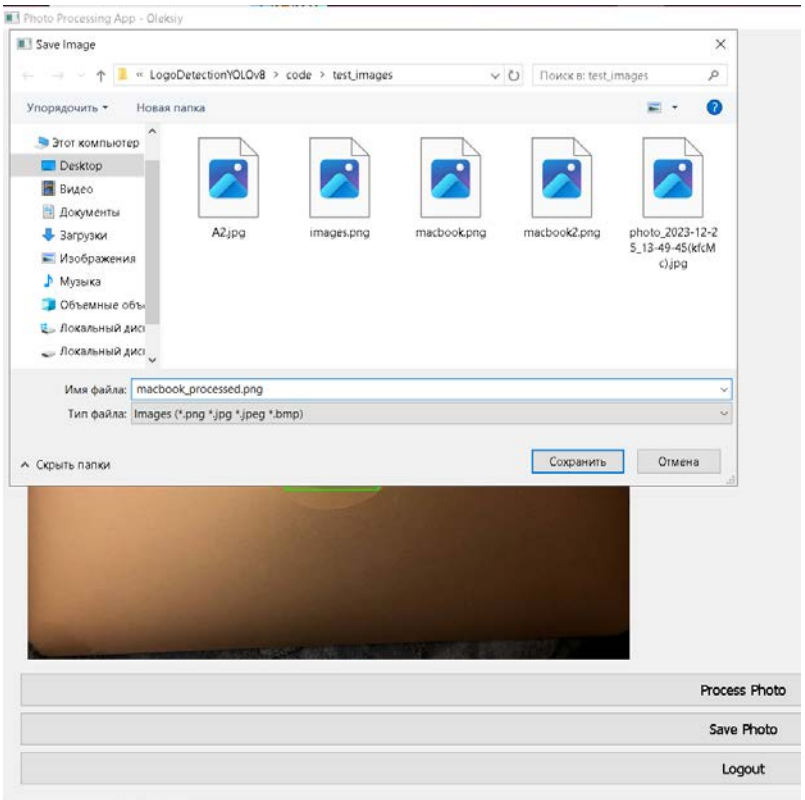


Рисунок 3.4 – процес збереження

Оброблене фото збережеться у обрану директорію. Результат представлено на рисунку 3.5.

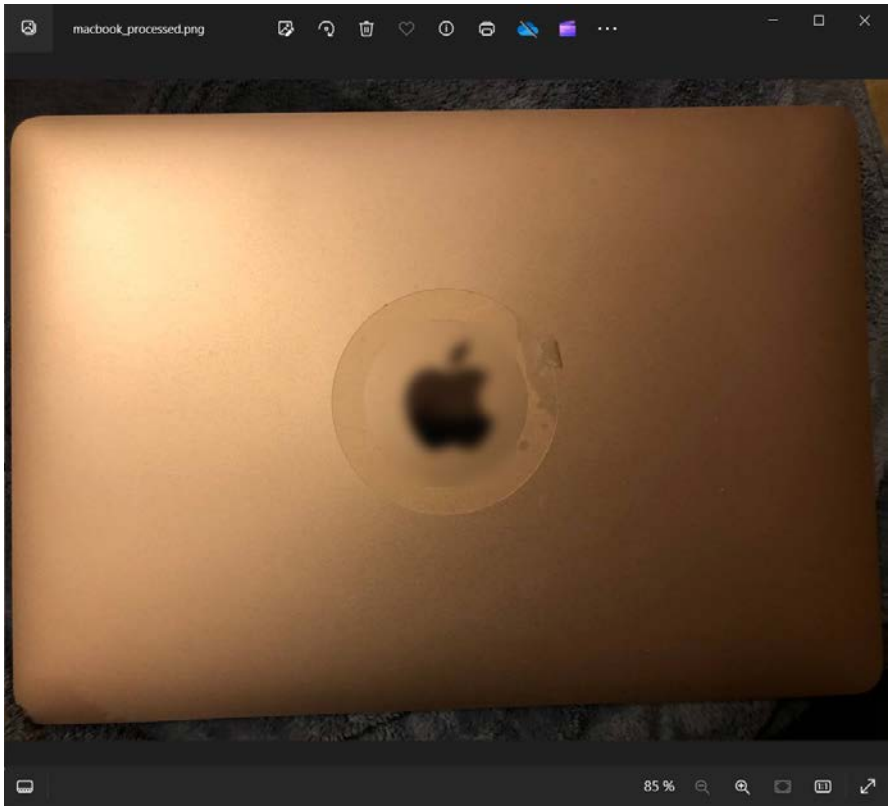


Рисунок 3.5 – результат обробки

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ЗАСТОСУНОК ДЛЯ ВИЯВЛЕННЯ ЛОГОТИПІВ БРЕНДІВ НА ФОТО НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

Графічний матеріал

КПІ.ІТ-0125.045480.06.99

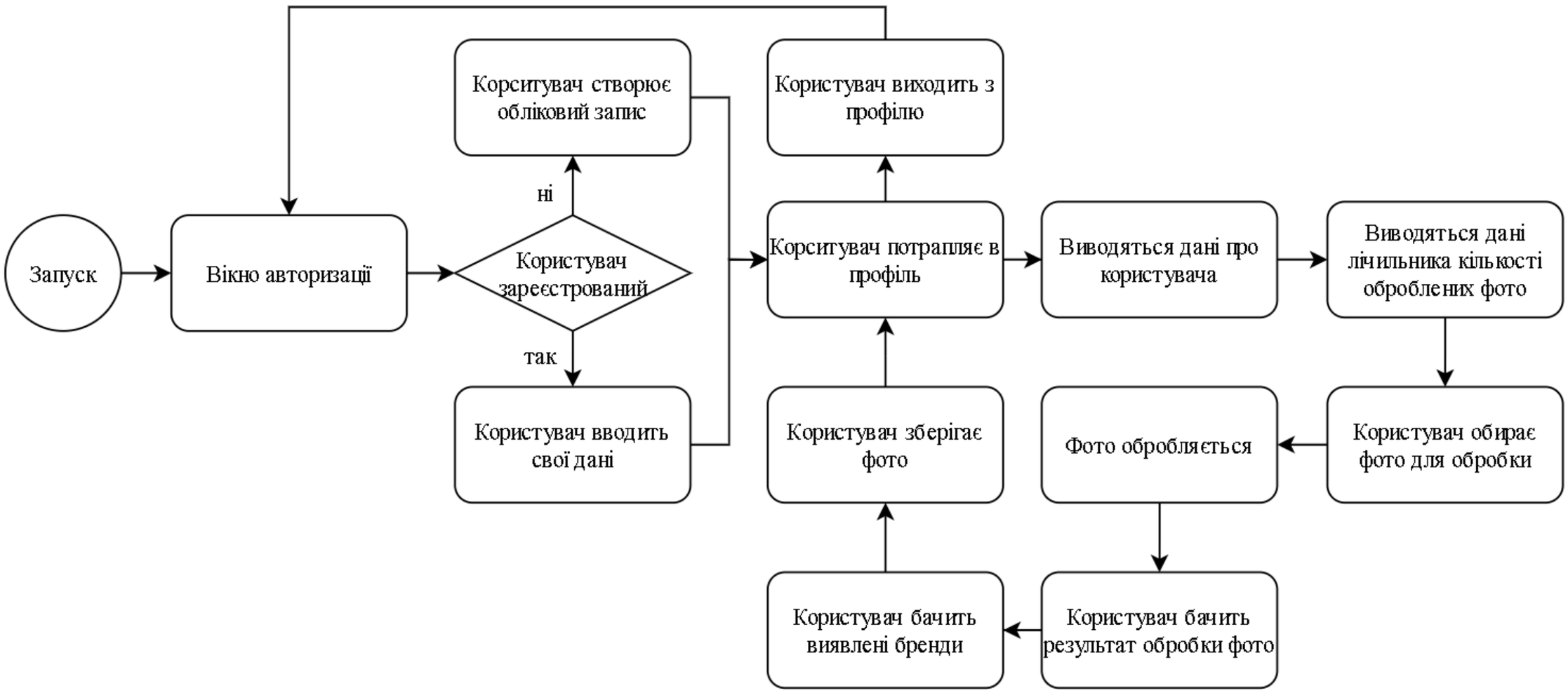
“ПОГОДЖЕНО”
Керівник проєкту:
_____ Валерій НОВІНСЬКИЙ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____Антон ТЕРЕШПОЛЬСЬКИЙ

Київ – 2024

СХЕМА БІЗНЕС ПРОЦЕСІВ ЗАСТОСУНКУ



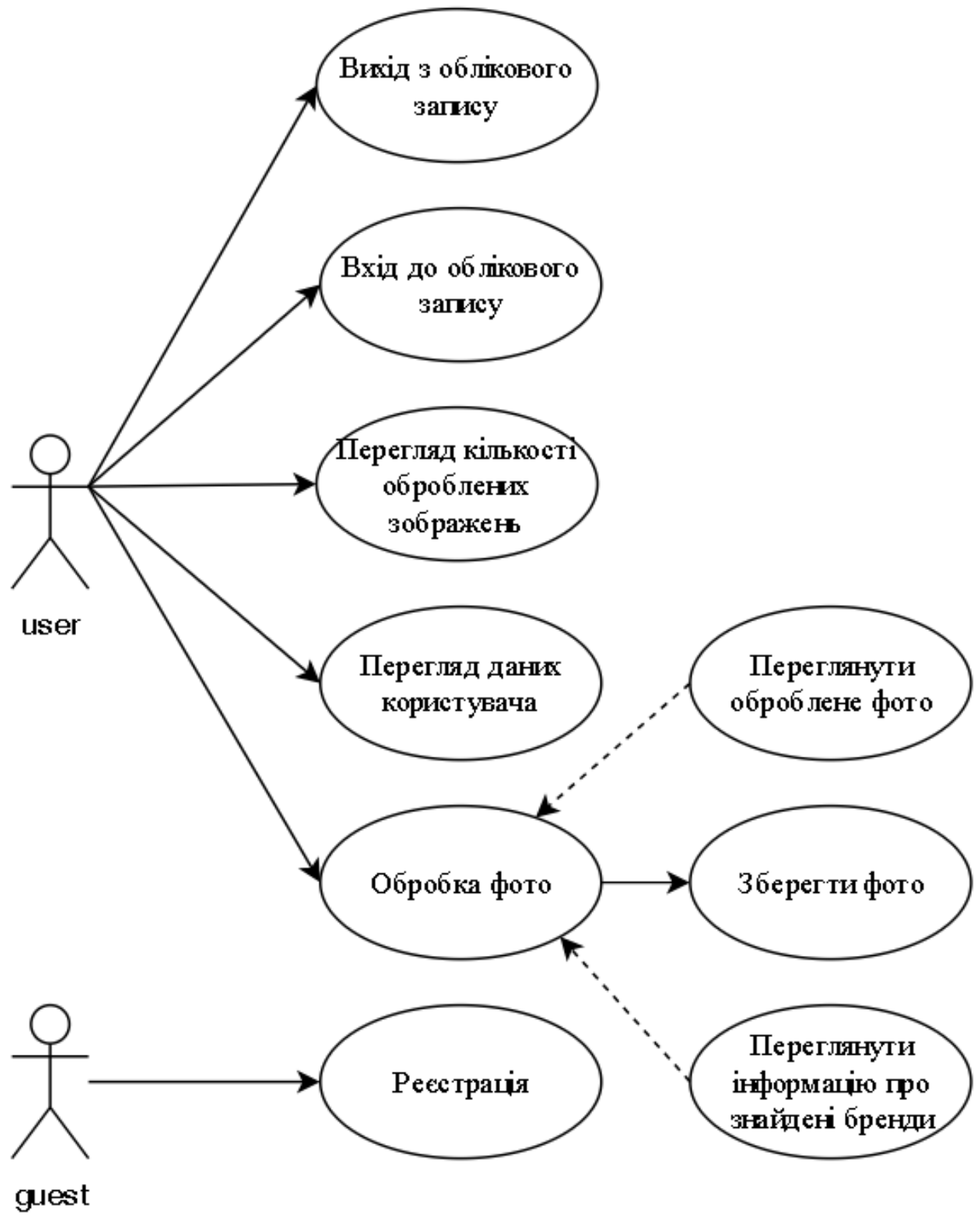
Демонстраційний плакат №1 до дипломного проекту

Застосунок для виявлення логотипів брендів на фото на основі штучного інтелекту

Виконав студент гр. ІТ-01 Терешпольський А. І.

Керівник Новінський Валерій Петрович

СХЕМА СТРУКТУРНА ВАРІАНТІВ ВИКОРИСТАННЯ



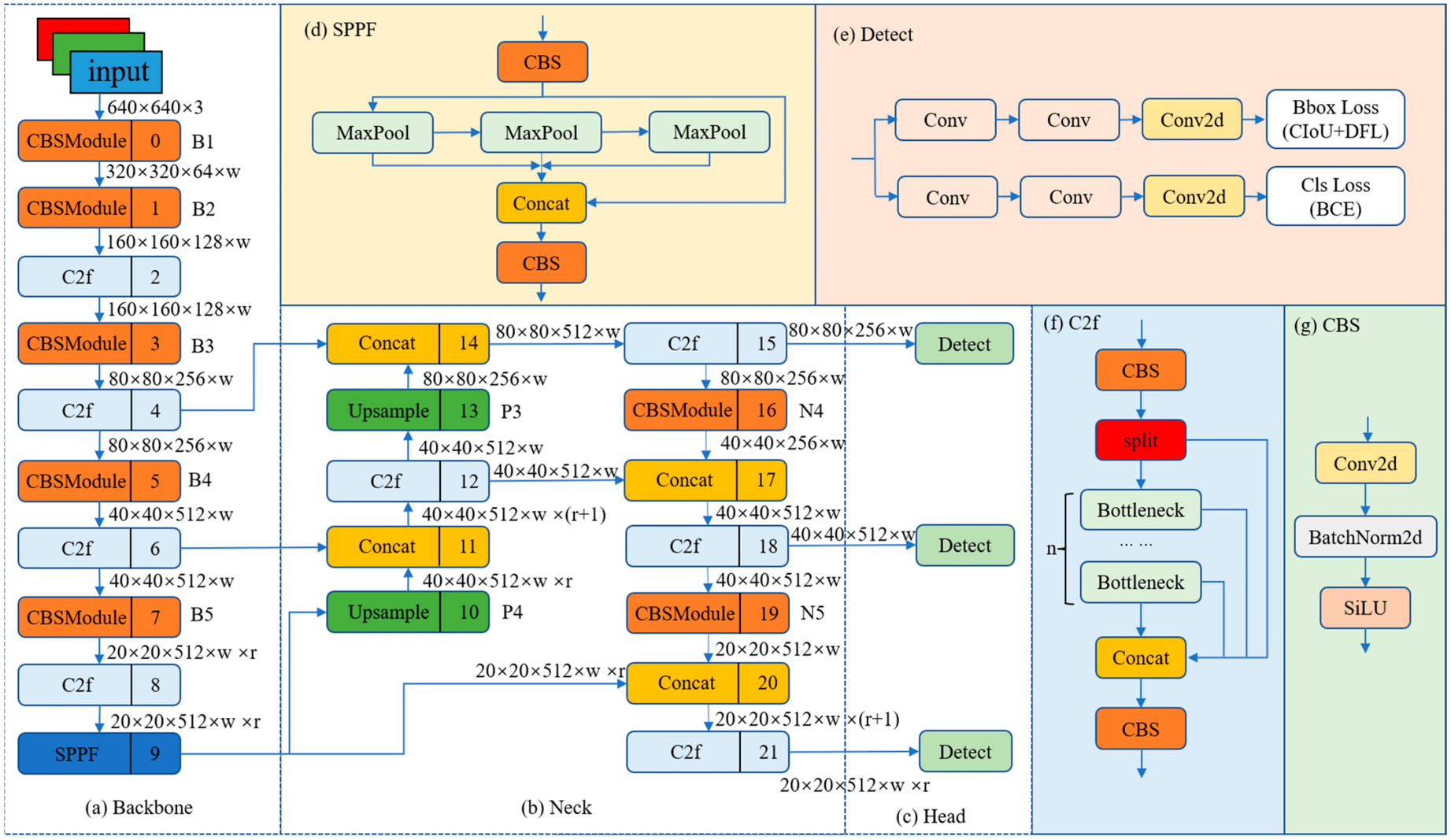
Креслення №1 до дипломного проекту

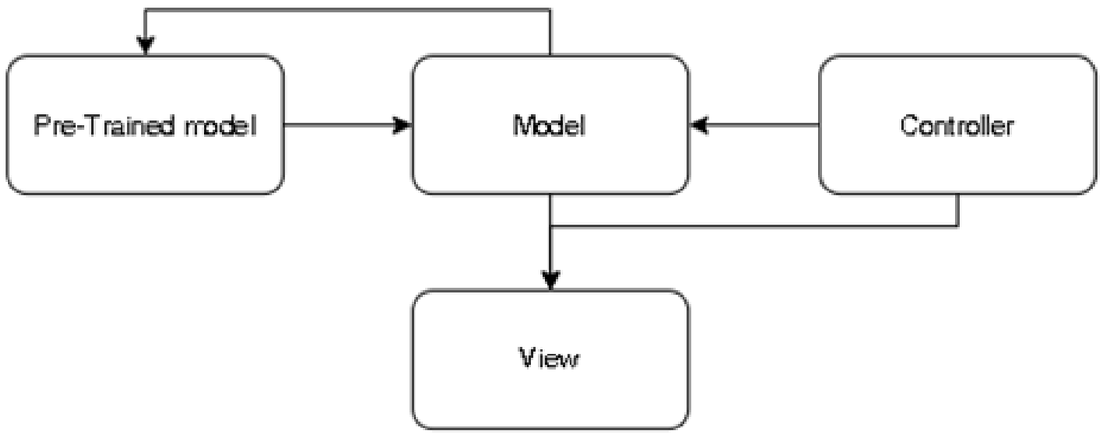
Застосунок для виявлення логотипів брендів на фото на основі штучного інтелекту

Виконав студент гр. ІТ-01 Терешпольський А. І.

Керівник Новінський Валерій Петрович

АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА МОДЕЛІ





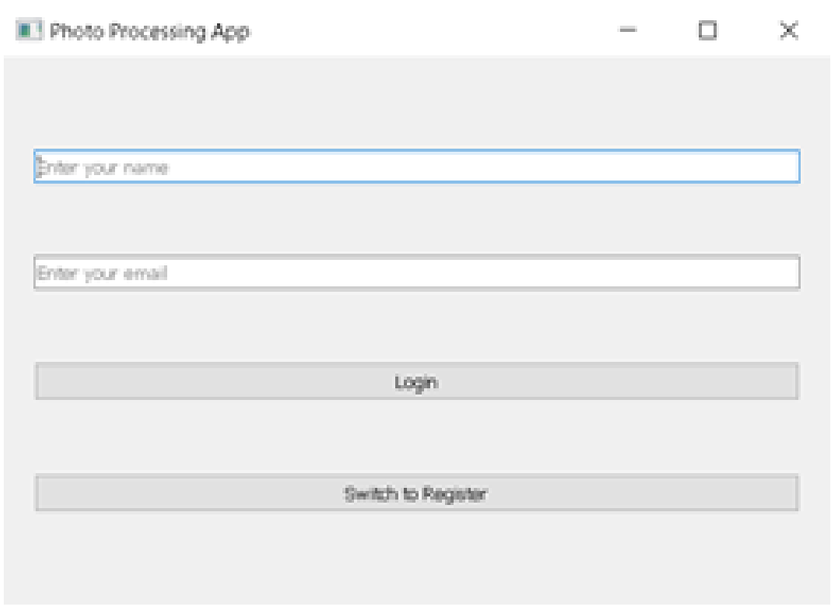
Демонстраційний плакат №2 до дипломного проекту

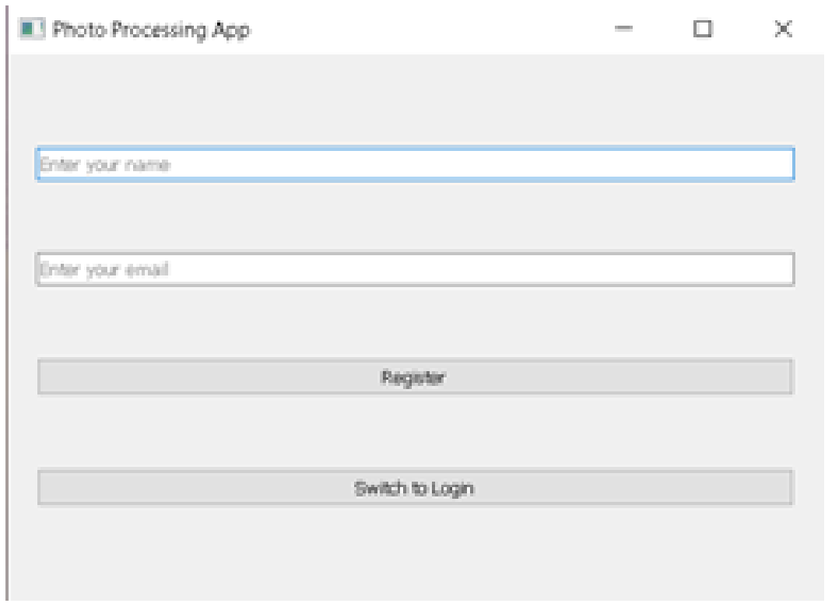
Застосунок для виявлення логотипів брендів на фото на основі штучного інтелекту

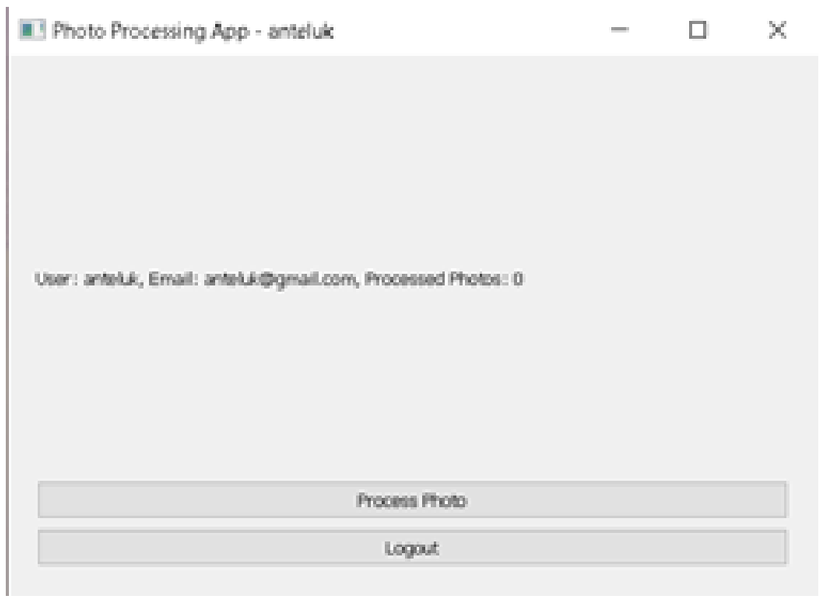
Виконав студент гр. ІТ-01 Терешпольський А. І.

Керівник Новінський Валерій Петрович

КРЕСЛЕННЯ ВИГЛЯДУ ЕКРАННИХ ФОРМ







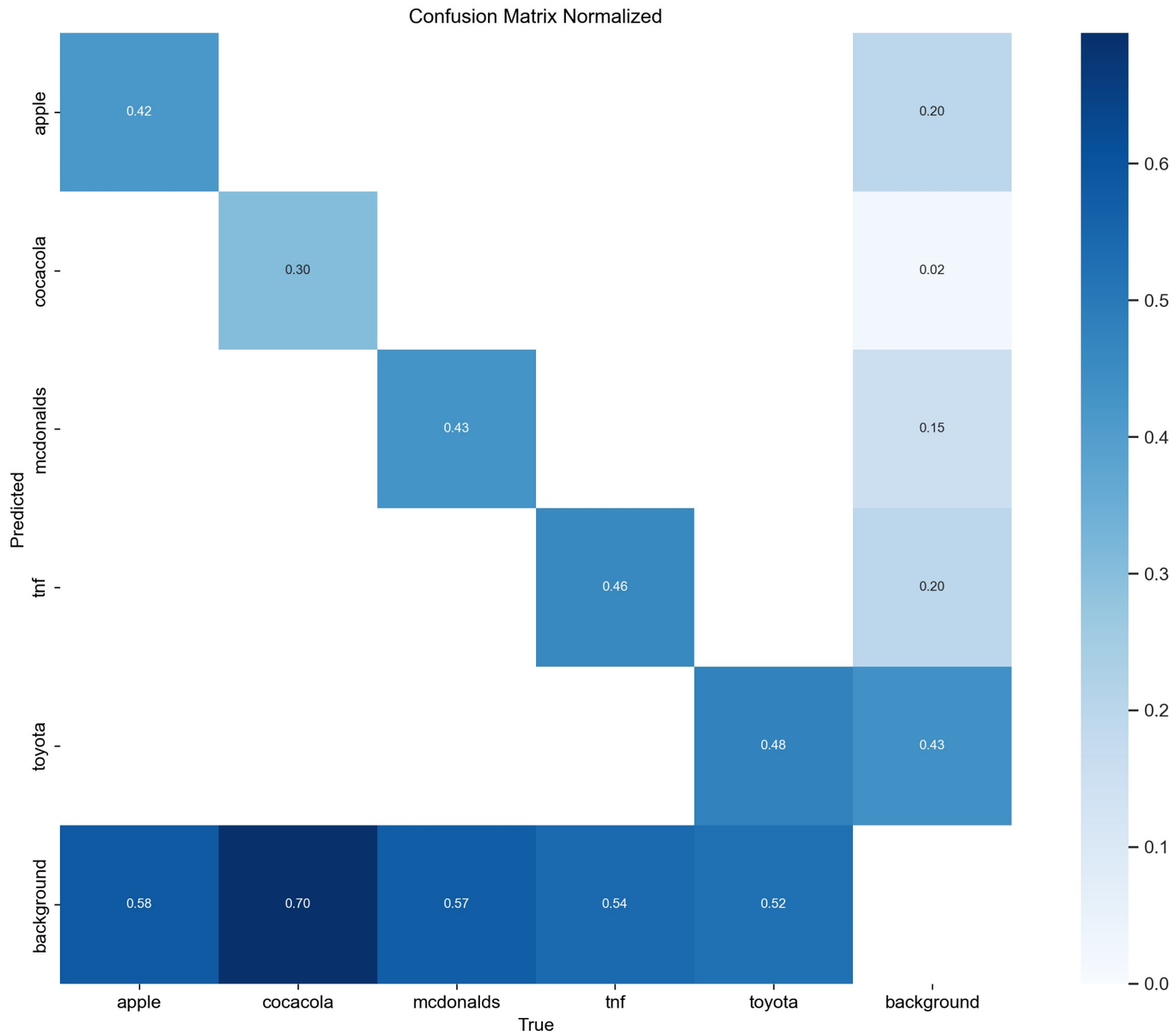
Креслення №2 до дипломного проекту

Застосунок для виявлення логотипів брендів на фото на основі штучного інтелекту

Виконав студент гр. ІТ-01 Терешпольський А. І.

Керівник Новінський Валерій Петрович

МАТРИЦЯ ПЛУТАНИНИ МОДЕЛІ



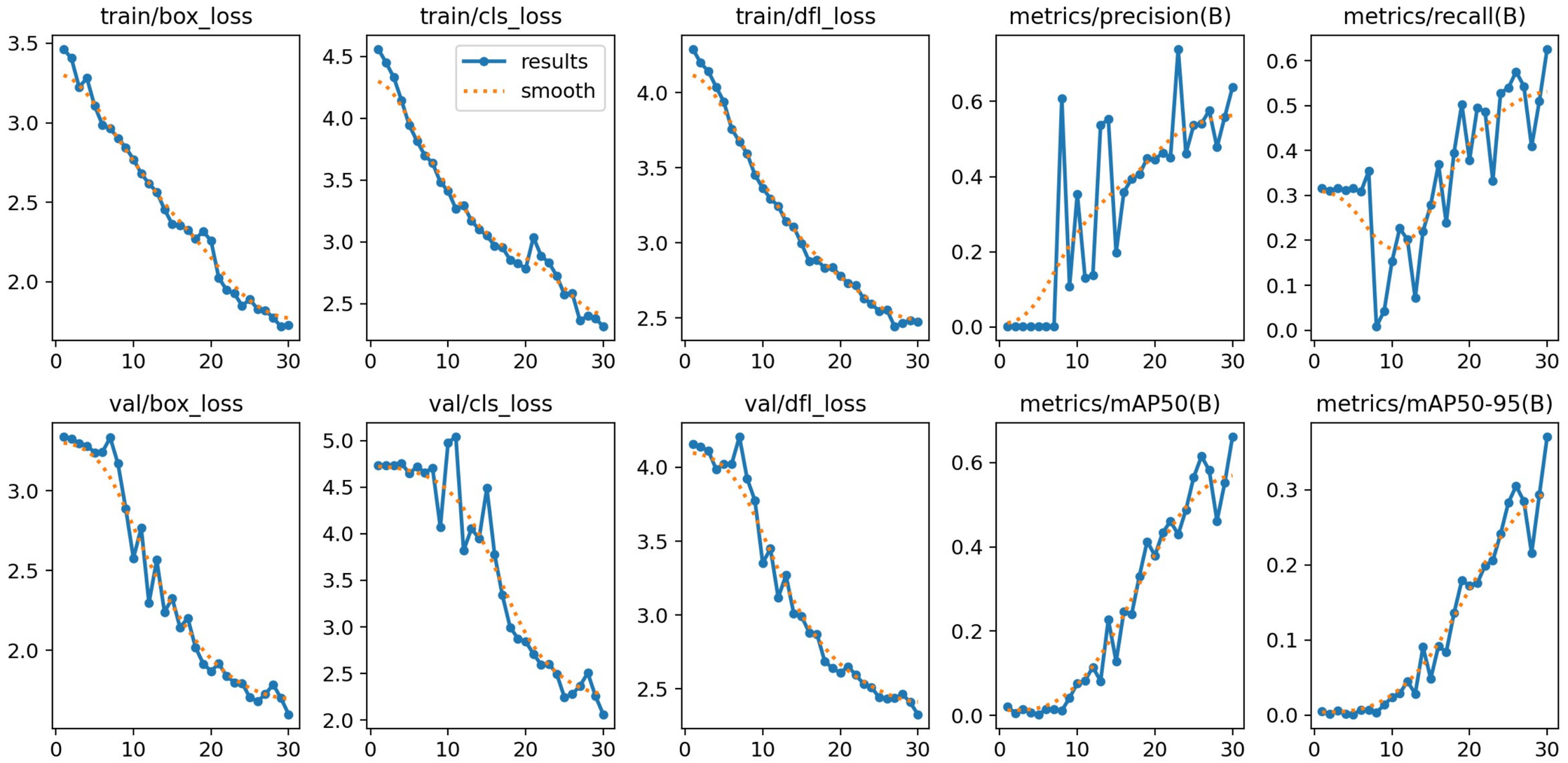
Демонстраційний плакат №3 до дипломного проекту

Застосунок для виявлення логотипів брендів на фото на основі штучного інтелекту

Виконав студент гр. ІТ-01 Терешпольський А. І.

Керівник Новінський Валерій Петрович

ГРАФІКИ ПАРАМЕТРІВ МОДЕЛІ



Демонстраційний плакат №4 до дипломного проекту

Застосунок для виявлення логотипів брендів на фото на основі штучного інтелекту

Виконав студент гр. ІТ-01 Терешпольський А. І.

Керівник Новінський Валерій Петрович

UNICHECK
by Turnitin

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1016326550

Дата перевірки:
06.06.2024 09:21:27 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
06.06.2024 09:52:06 EEST

ID користувача:
76913

Назва документа: **IT-01_Терешпольський_ПЗ**

Кількість сторінок: **42** Кількість слів: **4811** Кількість символів: **40846** Розмір файлу: **3.03 MB** ID файлу: **1016125447**

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

11.7%
Схожість

Найбільша схожість: **5.55%** з джерелом з Бібліотеки (ID файлу: **1016100816**)

2.99% Джерела з Інтернету	130	Сторінка 44
11.5% Джерела з Бібліотеки	269	Сторінка 45

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	2
Підозріле форматування	7 сторінок