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)
"___"_____2023 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Програмне забезпечення для розпізнавання наявності аксесуарів на обличчі людини

Виконав студент IV курсу, групи ІТ-93
(шифр групи)

Григоренко Ярослав Сергійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник ст. викл., Халус О. А. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації ст. викл., Головченко М. М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент кафедри ІСТ, к.т.н., доц., Шимкович В.М. _____
(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 –2023

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 комп'ютерних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2023 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Григоренку Ярославу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Програмне забезпечення для розпізнавання наявності аксесуарів на обличчі людини

керівник проєкту Халус Олена Андріївна, ст. викл.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «31»травня 2023 р. №2102-с

2. Термін подання студентом проєкту « 17 » червня 2023 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих програмних продуктів, постановка задачі.
2) Моделювання та конструювання програмного забезпечення: моделювання бізнес-процесів, проєктування архітектури та бази даних програмного забезпечення, конструювання продукту, аналіз безпеки даних.
3) Аналіз якості та тестування програмного забезпечення: аналіз якості кодової бази, тестування програмного забезпечення.
4) Впровадження та супровід програмного забезпечення: розгортання та підтримка програмного забезпечення.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема структурна послідовності

3) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «10» березня 2023 року _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	10.03.2023	
2	Аналіз існуючих методів розв’язання задачі	10.03.2023	
3	Постановка та формалізація задачі	21.03.2023	
4	Розробка інформаційного забезпечення	23.03.2023	
5	Алгоритмізація задачі	30.03.2023	
6	Обґрунтування вибору використаних технічних засобів	05.04..2023	
7	Розробка програмного забезпечення	08.04.2023	
8	Налагодження програми	21.04.2023	
9	Виконання графічних документів	12.05.2023	
10	Оформлення пояснювальної записки	16.05.2023	
11	Подання ДП на попередній захист	18.05.2023	
12	Подання ДП рецензенту	10.06.2023	
13	Подання ДП на основний захист	17.06.2023	

Студент _____ Ярослав ГРИГОРЕНКО

(підпис) (ініціали, прізвище)

Керівник _____ Олена ХАЛУС

(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 26 таблиць, 22 рисунків та 13 джерел – загалом 64 сторінки.

Дипломний проєкт присвячений розробці програмного забезпечення для розпізнавання наявності аксесуарів на обличчі людини

Мета: покращення безпеки, за допомогою відстеження в медичних та державних установах визначення наявності аксесуарів на обличчі людини на основі нейронної мережі

Об'єкт дослідження: процес розпізнавання наявності аксесуарів на обличчі людини.

Предмет дослідження: програмне забезпечення, яке використовується для розпізнавання наявності аксесуарів на обличчі людини.

У розділі аналізу вимог до програмного забезпечення надано докладний опис предметної області та сформульовано функціональні та нефункціональні вимоги до проєкту.

У розділі моделювання та конструювання програмного забезпечення визначено бізнес-процеси, розроблено архітектуру програмного забезпечення та проведено аналіз безпеки даних продукту.

У розділі аналізу якості та тестування програмного забезпечення проведено оцінку якості кодової бази та здійснено загальне тестування програмного забезпечення.

У розділі впровадження та супроводу програмного забезпечення описано процеси підтримки та розгортання програмного продукту.

КЛЮЧОВІ СЛОВА: ДЕСКТОПНИЙ ДОДАТОК, МАСКА, НЕЙРОННА МЕРЕЖА, БЕЗПЕКА, РОЗПІЗНАВАННЯ, КОВІД-19, ОБЛИЧЧЯ.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 26 tables, 22 figures and 13 sources – in total 64 pages.

The diploma project is dedicated to the development of software for the recognition of accessories on human faces.

Objective: to improve security by tracking the presence of accessories on human faces in medical and governmental institutions based on a neural network.

Research object: the process of recognizing the presence of accessories on human faces.

Research subject: software used for the recognition of accessories on human faces.

The analysis of software requirements section provides a detailed description of the subject area and formulates functional and non-functional requirements for the project.

The modeling and construction of software section defines business processes, develops the architecture of the software, and conducts data security analysis of the product.

The analysis of software quality and testing section evaluates the quality of the codebase and performs general testing of the software.

The implementation and maintenance of software section describes the support and deployment processes of the software product.

KEYWORDS: DESKTOP APPLICATION, MASK, NEURAL NETWORK, SECURITY, RECOGNITION, COVID-19, FACES.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-9308.045490.01.91	Технічне завдання	15	
3	А4	КПІ.ІП-9308.045490.02.81	Пояснювальна записка	64	
4	А4	КПІ.ІП-9308.045490.03.12	Текст програми	13	
5	А4	КПІ.ІП-9308.045490.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІП-9308.045490.05.34	Керівництво користувача	7	
7	А3	КПІ.ІП-9308.045490.06.99	Схема структурна варіантів	1	
			використань		
8	А3	КПІ.ІП-9308.045490.07.99	Схема структурна послідовностей	1	
9	А3	КПІ.ІП-9308.045490.08.99	Креслення вигляду екранних форм	1	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІП-9308.045490.00.90		
Розроб.		Григоренко Я.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.		Халус О. А.					
Н.контр.		Головченко М. М.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-93	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ НАЯВНОСТІ АКСЕСУАРІВ НА ОБЛИЧЧІ ЛЮДИНИ

Технічне завдання

КПІ.ІП-9308.045490.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Ярослав ГРИГОРЕНКО

Київ – 2023

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ НАЯВНОСТІ АКСЕСУАРІВ НА ОБЛИЧЧІ ЛЮДИНИ.

Галузь застосування:

Наведене технічне завдання поширюється на розробку програмного забезпечення «Програмне забезпечення для розпізнавання наявності аксесуарів на обличчі людини» КПІ.ІП-9308.045490.01.91, котре використовується для розпізнавання наявності аксесуарів на обличчі людини та призначена для охоронних служб, систем безпеки та контролю доступу.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки програмного забезпечення для розпізнавання наявності аксесуарів на обличчі людини 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 автоматичного виявлення та аналізу аксесуарів.

Метою розробки є покращення безпеки, за допомогою відстеження в медичних та державних установах визначення наявності аксесуарів на обличчі людини на основі нейронної мережі.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

– форма реєстрації користувача (Рисунок 4.1);

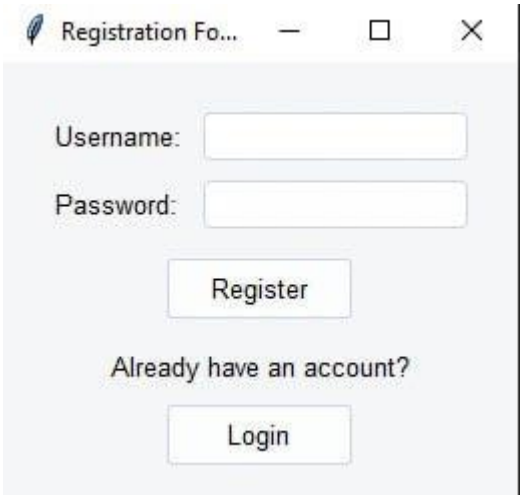

Рисунок 4.1 – Форма реєстрації користувача

– форма авторизації користувача (Рисунок 4.2);

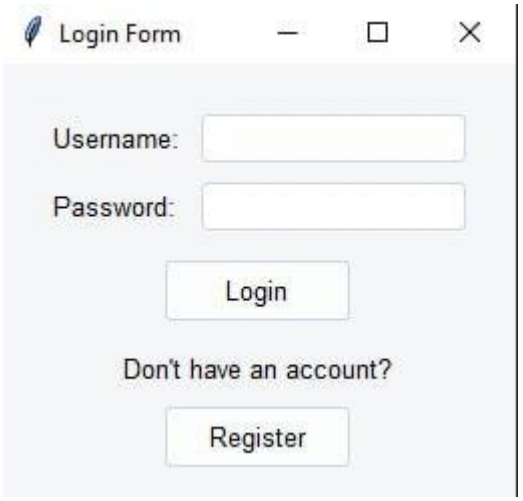

Рисунок 4.2 – Форма авторизації користувача

– головна сторінка (Рисунок 4.3);

Рисунок 4.3 – Головна сторінка

– перегляд статистики (Рисунок 4.4) та перегляд збережених зображень (Рисунок 4.5);

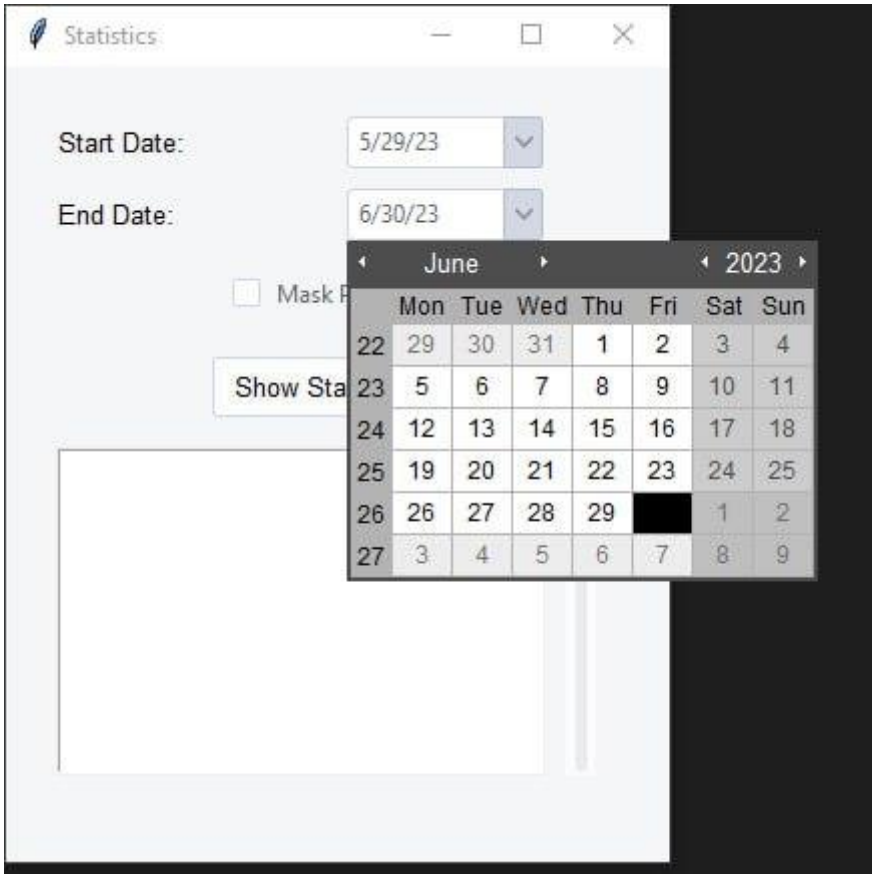

Рисунок 4.4 – Перегляд статистики

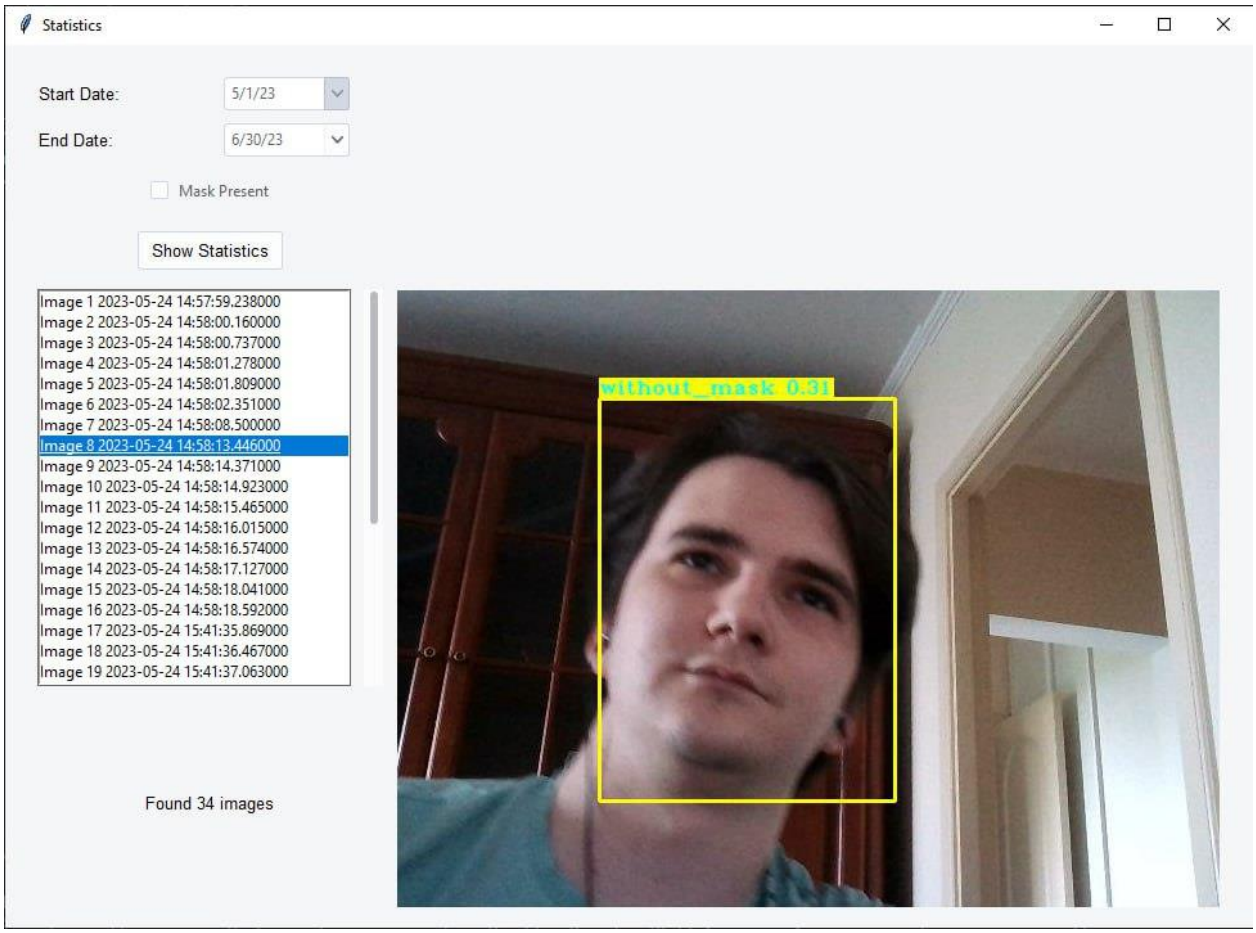

Рисунок 4.5 – Перегляд збережених зображень

– перегляд відеопотоку (Рисунок 4.6);

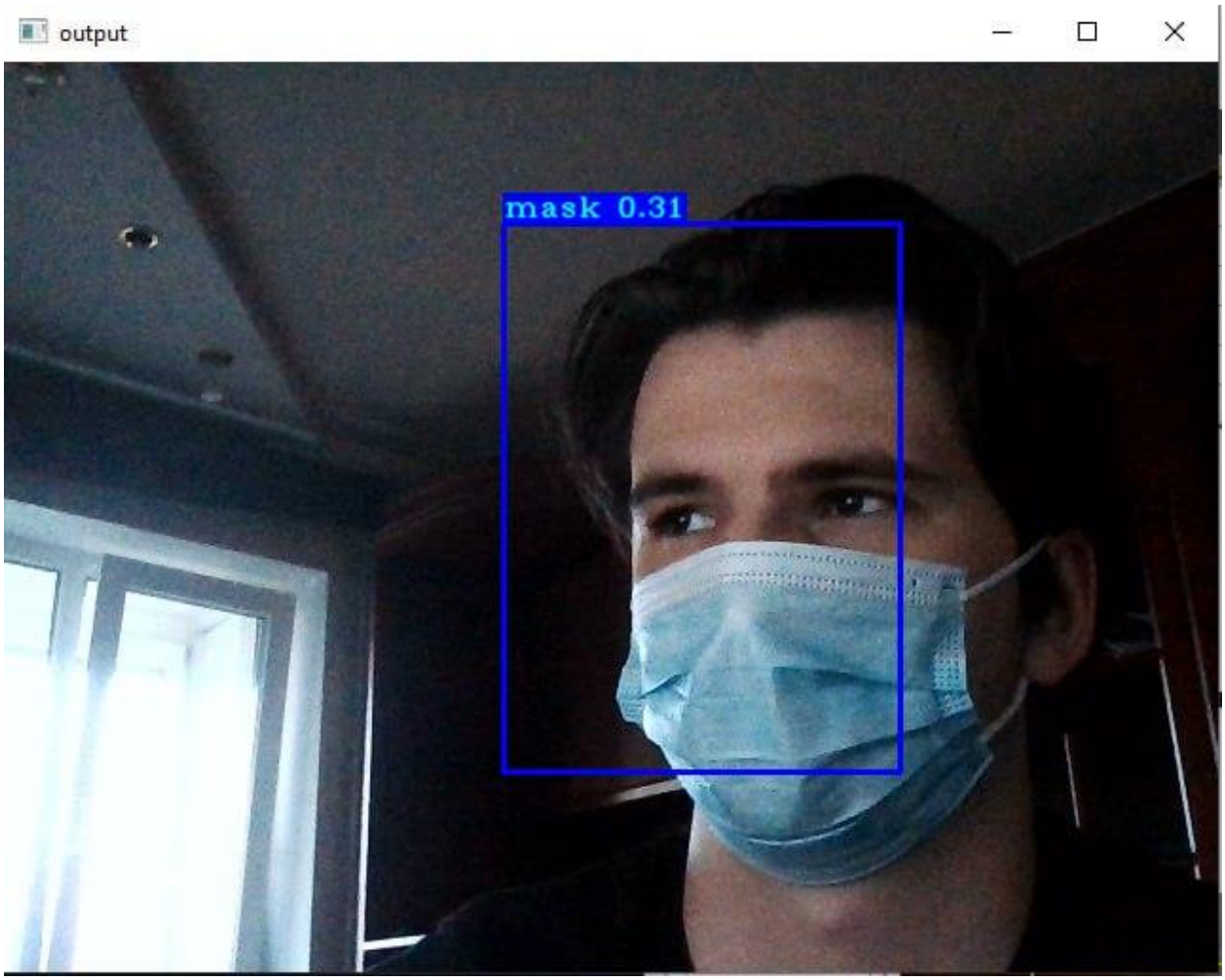

Рисунок 4.6 – Перегляд відеопотоку

– сторінка адміністратора (Рисунок 4.7);

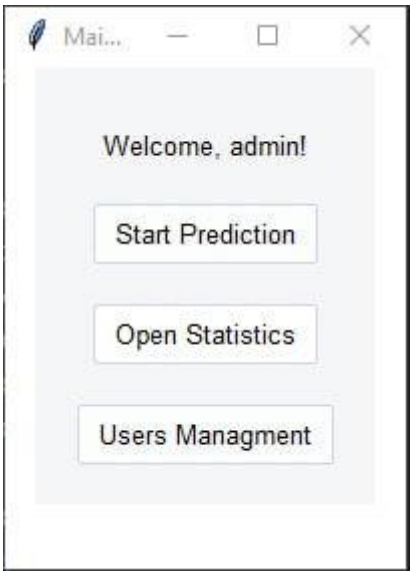

Рисунок 4.7 – Сторінка адміністратора

– список користувачів (Рисунок 4.8.).

Рисунок 4.8 – Список користувачів

4.1.2 Для користувача:

– авторизація;
– реєстрація;
– перегляд статистики;
– перегляд відео з камери в режимі реального часу;
– перегляд збережених зображень.

4.1.3 Для адміністратора системи:

– авторизація;
– перегляд списку користувачів;
– редагування списку користувачів
– перегляд статистики;
– перегляд відео з камери в режимі реального часу;
– перегляд збережених зображень.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до 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 персональних комп‘ютерах.

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i3;
– об‘єм RAM: 4 Гб;
– об‘єм VRAM: 4 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт (за умови використання хмарної бази даних).

Рекомендована конфігурація технічних засобів:

– тип процесору: AMD Ryzen 5 9500X;
– об‘єм RAM: 128 Гб;
– об‘єм VRAM: 16 Гб;
– підключення до мережі Інтернет зі швидкістю від 1000 мегабіт (за умови використання хмарної бази даних).

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства Windows.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі відеопотоку.

4.5.2 Вимоги до вихідних даних

Вимоги до вихідних даних не висуваються.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Python.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі Windows VSCode.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді текстових файлів.

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

Згенерувати інсталяційну версію програмного забезпечення.

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 структурна варіантів використання;

– схема структурна послідовності;

– креслення вигляду екранних форм.

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 рекомендованої літератури	10.03	
2.	Аналіз існуючих методів розв’язання задачі	21.03	
3.	Постановка та формалізація задачі	23.03	Мета розробки
4.	Розробка інформаційного забезпечення	30.03	Фізична модель бази даних
5.	Алгоритмізація задачі	05.04	Схема архітектури програмного забезпечення та діаграма класів
6.	Обґрунтування вибору використаних технічних засобів	08.04	
7.	Розробка програмного забезпечення	21.04	Тексти програмного забезпечення
8.	Налагодження програми	12.05	Тести, результати тестування
9.	Виконання графічних документів	16.05	Графічний матеріал проекту
10.	Оформлення пояснювальної записки	18.05	Пояснювальна записка

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

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

Пояснювальна записка
до дипломного проєкту

на тему: Програмне забезпечення для розпізнавання наявності аксесуарів на обличчі людини

КПІ.ІП-9308.045490.02.81

Київ – 2023

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

COVID-19	–	Coronavirus disease 2019 – коронавірусна хвороба 2019
ВООЗ	–	Всесвітня організація охорони здоров'я
YOLO	–	You Only Look Once
IT	–	Інформаційні технології
SSD	–	Single Shot MultiBox Detector
GUI	–	Graphical user interface
БД	–	База даних.
VSCode	–	Visual Studio Code
IDE	–	Integrated Development Environment

ВСТУП

За останні кілька років маски стали необхідним елементом захисту від вірусів, особливо під час пандемії COVID-19. Їх ефективність полягає в тому, що вони запобігають розповсюдженню інфекції. Однак, оскільки багато людей може бути носіями вірусу без симптомів, розумною стратегією є використання масок всіма особами у громадських місцях та на вулиці. Коректне носіння масок, особливо у поєднанні з вакцинацією, є ефективним способом зниження поширення COVID-19, включаючи нові варіанти вірусу.

Проте виникла потреба в ефективному способі контролювання використання масок в громадських місцях. Традиційні методи, такі як найм працівників для контролю, мають свої недоліки, такі як збільшення ризику для здоров'я працівників, додаткові контакти з людьми та складнощі у забезпеченні достатньої кількості персоналу. Однак, сучасні технології можуть допомогти вирішити цю проблему.

Шляхом встановлення відеокамери та її підключення до сервера можна використовувати нейронні мережі для відслідковування наявності медичних масок на обличчях людей. Це дозволяє автоматично визначати, чи дотримуються люди правила щодо носіння масок. Такий підхід дозволяє зменшити людський фактор та забезпечити надійність контролю. Завдяки використанню нейронних мереж із відеоданими, система може швидко та ефективно реагувати на виявлення відсутності захисних масок, що дозволяє забезпечити безпеку та спокій у громадських місцях.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Коронавіруси - це велика група вірусів, які можуть спричиняти захворювання у людей та тварин. Один із найвідоміших коронавірусів, що став причиною глобальної пандемії, називається SARS-CoV-2, а захворювання, яке він викликає, отримало назву COVID-19[1].

Зараження коронавірусом по різних країнах Європи на 18.05.2023

Країна	Всього інфі-кована	Смер-тельні випадки	Виду-жали	Наразі хворіють
Франція	40054863 +4149	167052	39807374 +6609	80437
Німеччина	38421224 +1075	173941 +54	38225300 +1600	21983
Італія	25828252	190080	25511697	126475
Велика Британія	24569895	223396	24346499	
Росія	22900755	398736	22323212	178807
Іспанія	13845825	120964	13681014	43847
Нідерланди	8610372	22992	8587380	
Польща	6516030 +140	119595 +1	5335940	1060495
Австрія	6073496 +364	22488 +2	6044358 +362	6650
Греція	6044517 +18741	36863 +52	5985866 +2067	21788
Португалія	5584388 +1401	26668 +52	N/A	N/A
Україна	5538357	112210	5417249	8898

Рисунок 1.1 - Статистика зараження коронавірусом по країнах Європи

На Рисунку 1.1 наведено статистику зараження коронавірусом по країнах Європи станом на 18.05.2023[2].

Щоб захиститись від COVID-19 і зменшити ризик зараження, рекомендується дотримуватись наступних заходів захисту[3]:

– Уникати контакту з потенційно контамінованими вірусами поверхнями, зокрема тваринами, у регіоні із виявленими випадками інфікування.

– Ретельно та часто мити руки з милом та обробляти їх антисептиком.

– Уникати скупчення людей.

– При підозрі або виявленні хвороби, залишатись вдома та звернутись до лікаря.

– Використовувати захисну маску.

– Носіння масок має важливе значення для контролю поширення COVID-19. Основні причини, чому маски рекомендуються наведені нижче:

– Захист від крапельно-інфекційного шляху передачі. Маски можуть допомогти запобігти поширенню вірусу, якщо хтось, хто є носієм, кашляє, чхає або розмовляє. Вони затримують краплі, які містять вірус, і зменшують ризик передачі інфекції іншим людям.

– Носіння маски може знизити ризик зараження вірусом, оскільки вона може перешкодити потраплянню в дихальні шляхи крапель, які містять вірус.

– Попередження зараження вірусом від людей без симптомів. Багато людей інфікується COVID-19, навіть якщо не виявляють симптомів. Маски можуть допомогти зменшити ризик передачі вірусу від таких осіб, навіть у випадку, якщо носій не знає про своє зараження.

– Зменшення ризику контакту з поверхнею рота та носа. Носіння маски може нагадувати про небезпеку частого дотику рота та носа, що може бути шляхом введення вірусу в організм.

– Згідно з рекомендаціями ВООЗ, маски слід носити в наступних випадках:

– У переповнених, закритих або погано провітрюваних приміщеннях;

– Якщо ви вважаєте, що у вас може бути COVID-19, коли ви перебуваєте у спільному просторі з іншими;

– Коли ви перебуваєте у спільному просторі з людиною, яка має симптоми COVID-19 або є позитивною на COVID-19;

– Коли ви перебуваєте у громадському просторі, і ви маєте високий ризик серйозно захворіти на COVID-19 або померти.[4]

Слідкування за наявністю масок на обличчі людей вручну є складним завданням, яке вимагає значних зусиль та додаткового персоналу. Отже, є важливим створення системи, яка автоматично відстежуватиме наявність масок та забезпечуватиме статистику, яка сприятиме боротьбі з цією хворобою та збереженню життя людей.

1.2 Змістовний опис і аналіз предметної області

Програмне забезпечення для розпізнавання наявності аксесуарів на обличчі людини, у тому числі захисних масок є важливим з точки зору контролю і профілактики поширення COVID-19. Таке ПЗ має потенціал допомогти автоматично виявляти, чи носить людина маску на обличчі на основі зображень або відео. ПЗ може бути побудоване на основі нейромережі.

Аналізуючи цю предметну область, можна відзначити декілька ключових аспектів. Для розробки такої нейромережі необхідна велика кількість маркованих даних, тобто зображень або відео з позначками "маска" або "без маски". Ці дані можуть бути зібрані шляхом ручного позначення зображень або використанням автоматичних алгоритмів виявлення масок.

Алгоритм нейромережі для визначення маски на обличчі може базуватись на методах комп'ютерного зору, використовуючи глибокі нейронні мережі, зокрема засновані на згорткових нейронних мережах. Ці мережі можуть вивчати різноманітні особливості і зразки, що характеризують наявність маски на обличчі та робити прогнози на основі цих ознак.

Також, для ефективної роботи нейромережі визначення маски в реальному часі, її необхідно оптимізувати щодо швидкості та точності. Це може бути досягнуто шляхом оптимізації архітектури мережі, використання спеціалізованих алгоритмів навчання та оптимізації параметрів моделі.

Застосування такої нейромережі може бути різноманітним. Наприклад, вона може бути використана для автоматичного контролю дотримання правил носіння масок у громадських місцях або на входах до будівель, або може бути використане для збирання статистики по носінню аксесуарів. Також, вона може знайти застосування в системах відеоспостереження, де може бути важливо виявляти особи без масок та приймати відповідні заходи. Ще одна сфера використання – охоронна система. В сфері безпеки, таке програмне забезпечення може використовуватися в системах відеоспостереження для автоматичного виявлення осіб, які намагаються сховати свою ідентичність за допомогою аксесуарів. Наприклад, в об'єктах загального користування, таких як аеропорти, вокзали або банки, це програмне забезпечення може сприяти виявленню потенційно небезпечних осіб або злочинців, які намагаються приховати свою обличчя. За допомогою системи сповіщень або автоматичних реакцій, охоронні служби можуть оперативно реагувати на такі ситуації та запобігати можливим загрозам.

Загалом, розробка нейромережі для визначення наявності аксесуарів на обличчі людини, у тому числі медичної маски, має значний потенціал у боротьбі зі спалахом інфекційних захворювань та сприяє підвищенню рівня безпеки в громадських місцях.

Дане рішення і є метою дипломного проєкту.

1.3 Аналіз існуючих технологій та успішних ІТ-проєктів

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації для розпізнавання наявності аксесуарів на обличчі людини, а саме медичної маски. Далі будуть

розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.3.1 Аналіз відомих алгоритмічних та технічних рішень

Існує декілька технічних рішень, які можна використати при розробці програмного забезпечення для розпізнавання наявності аксесуарів на обличчі людини.

Першим варто згадати одне із найпопулярнішим рішень - YOLO (You Only Look Once). Це потужна архітектура нейромережі для об'єктного визначення в реальному часі. Вона вперше була представлена в 2016 році і з тих пір стала одним із найпопулярніших методів в цій області. Основна ідея YOLO полягає в тому, що алгоритм одноразово пропускає зображення через мережу та визначає об'єкти відразу на всіх його частинах.

Архітектура YOLO використовує глибоку згорткову нейромережу, що складається зі стеку згорткових шарів, згорткових шарів з редукцією розміру та повнозв'язаних шарів. Ключовою особливістю цієї архітектури є використання одного останнього шару, що відповідає за виявлення об'єктів. Цей шар виробляє прогнози щодо класів об'єктів та прямокутних рамок, що охоплюють ці об'єкти. Оскільки він працює на всьому зображенні одразу, а не на окремих областях, YOLO має високу швидкодію та здатність виявляти об'єкти в режимі реального часу.

Для покращення точності та локалізації об'єктів, YOLO використовує алгоритм "anchor boxes" (якісні прямокутники). Цей алгоритм дозволяє встановити попередньо визначені форми та розміри об'єктів, що враховуються при виявленні. Це допомагає забезпечити більш точну локалізацію об'єктів на зображенні.

YOLO використовує передові техніки навчання нейромереж, такі як зворотнє поширення помилки (backpropagation) та оптимізація за допомогою стохастичного градієнтного спуску (stochastic gradient descent), для покращення якості виявлення об'єктів. Його можна навчати на великих

наборах даних з розміченими зображеннями, що містять об'єкти, і з часом він може набути високої точності та здатності розпізнавати широкий спектр об'єктів.

Застосування YOLO для виявлення наявності аксесуарів на обличчі людини у сфері захисту має великий потенціал. Це може бути використано в системах безпеки, контролі доступу до об'єктів, відеоспостереженні та багатьох інших сценаріях, де важлива ідентифікація людей та виявлення потенційно небезпечних ситуацій. Завдяки швидкодії та високій точності YOLO стає ефективним інструментом для покращення безпеки та захисту у різних сферах життя.

Ще одним рішенням є Single Shot MultiBox Detector, далі SSD. Це інноваційна архітектура нейромережі, розроблена для задач об'єктного визначення на зображеннях. Вона була вперше запропонована в 2016 році і відтоді стала одним із найефективніших методів для виявлення об'єктів у реальному часі.

Основною особливістю SSD є його здатність одночасно визначати об'єкти різних розмірів та класів на зображенні. Замість використання одного великого фіксованого вікна (як у багатьох інших архітектур), SSD використовує множину прямокутних вікон різних розмірів та співвідношень сторін. Кожне вікно має декілька відповідних класів, які модель визначає для об'єктів, що знаходяться в цій області[5].

Архітектура SSD складається з базової мережі, такої як VGG або ResNet, яка використовується для витягування високорівневих ознак з зображення, і декількох додаткових згорткових шарів, які використовуються для виявлення об'єктів різних масштабів. Ці додаткові шари допомагають моделі впевнитися, що об'єкти будуть виявлені незалежно від їхнього розміру.

SSD навчається за допомогою великого набору даних з розміченими зображеннями, де об'єкти мають відповідні класи та мітки. Завдяки цьому

навчанню, модель може визначати об'єкти різних класів на нових зображеннях.

Наступне проаналізоване рішення це MobileNet[6]. Вона є інноваційною архітектурою нейромережі, спеціально розробленою для розпізнавання об'єктів на пристроях з обмеженими ресурсами, таких як мобільні телефони та вбудовані системи. Її головна мета - забезпечити високу точність визначення об'єктів при низькій кількості параметрів та швидкості роботи.

Особливістю MobileNet є використання "групових згорток" (depthwise separable convolutions), що дозволяють ефективно виконувати операції згортки, мінімізуючи кількість параметрів та обчислювальний затрати. Замість використання повних згорток для кожного каналу вхідних даних, групові згортки розбивають канали на групи та застосовують згортку окремо для кожної групи. Це дозволяє значно зменшити кількість параметрів і зробити мережу легшою та швидшою.

MobileNet використовує глибокі згорткові шари для витягування ознак з вхідних зображень та має декілька блоків, що складаються з групових згорток, повних згорток та проміжних шарів активації. Ці блоки дозволяють моделі знаходити різноманітні ознаки різного рівня абстракції, поступово підвищуючи складність розпізнавання.

MobileNet є дуже популярною архітектурою для задач розпізнавання об'єктів на мобільних пристроях, оскільки вона демонструє високу ефективність при обмежених обчислювальних ресурсах. Вона знаходить широке застосування в мобільних додатках, системах розпізнавання образів, вбудованих системах та інших сферах, де важливо забезпечити високу точність розпізнавання об'єктів при низькій споживанні ресурсів.

Для дипломного проєкту вирішено обрати YOLOv4[7], адже це нейромережа побудована на основі архітектури YOLO. Вона є покращеною версією попередніх моделей YOLO, яка пропонує покращену точність та швидкодію. YOLOv4 пропонує кращу точність в порівнянні з попередніми

версіями YOLO, завдяки використанню різноманітних технік, таких як багатомасштабне навчання, додаткові шари згорткових мереж та оптимізації ваг моделі. Це дозволяє досягати високої точності визначення об'єктів різних класів. Крім того, YOLOv4 має підтримку для виявлення багатьох класів об'єктів, здатність роботи зі зображеннями високої роздільної здатності та можливість використовувати пристрої з обчислювальними прискорювачами, такими як GPU, для прискорення обчислень.

Також варто проаналізувати архітектури програмного забезпечення.

Монолітна архітектура – усі компоненти додатку (інтерфейс, бізнес-логіка, доступ до даних) знаходяться в одному додатку.

Клієнт-серверна архітектура – додаток складається з двох основних компонентів - клієнта, що забезпечує інтерфейс користувача, і сервера, що забезпечує бізнес-логіку та доступ до даних.

Компонентна архітектура – додаток розбитий на незалежні компоненти, які можуть виконувати специфічні функції та взаємодіяти між собою.

Розподілена архітектура – функціональність розподілена між різними вузлами або комп'ютерами, що взаємодіють між собою через мережу.

Архітектура на основі подій – додаток побудований на основі взаємодії між компонентами через події та підписку на них.

Багаторівнева архітектура – додаток розбитий на логічні шари, такі як інтерфейс, бізнес-логіка та доступ до даних, з кожним шаром відповідальним за певну функціональність.

Для дипломного проєкту вирішено обрати багаторівневу архітектуру, адже вона надає масштабованість, модульність, гнучкість та можливість повторного використання коду.

1.3.2 Аналіз допоміжних програмних засобів та засобів розробки

Для ефективної реалізації обраної архітектури та технічних рішень розумно оцінити доступні програмні інструменти, які полегшать процес розробки продукту.

Перш за все, варто обрати мову програмування. Найбільш популярні варіанти – Python, R та Java. Порівняння мов програмування для розробки нейронних мереж можна провести з різних аспектів, таких як широко поширеність, продуктивність, легкість використання, екосистема та підтримка громади розробників.

Python[8] є однією з найпопулярніших мов програмування для розробки нейронних мереж завдяки своїй простоті, легкості використання та широкій екосистемі бібліотек, таких як TensorFlow, Keras, PyTorch і scikit-learn. Він також має велику підтримку спільноти розробників і багато ресурсів для навчання.

R[9] є мовою програмування, спеціально розробленою для статистичного аналізу та машинного навчання. Вона має багатий набір пакетів для нейронних мереж, таких як neuralnet, kerasR та MXNetR. R має сильну статистичну підтримку і широке використання в академічних дослідженнях.

Java[10] є мовою програмування з великою базою розробників і широким використанням у великих проєктах. Для розробки нейронних мереж використовуються бібліотеки, такі як Deeplearning4j, DL4J і Encog. Java може бути корисним в випадках, коли важлива продуктивність та масштабованість системи.

Вирішено обрати мову Python, адже завдяки своїй багатофункціональності, розширюваності та простоті використання це чудовий. Також завдяки великій спільноті розробників, рішення будь-яких проблем знаходяться швидше та вивчення нового матеріалу відбувається ефективніше.

Для створення графічного інтерфейсу користувача також вирішено обрати Python, а саме Tkinter. Це потужний інструмент, який дозволяє розробникам створювати інтуїтивно зрозумілі та привабливі користувацькі інтерфейси для своїх програм. Tkinter базується на бібліотеці Tk, яка є надійним і стабільним фреймворком для розробки GUI.

Однією з основних переваг Tkinter є його простота використання. Вона має простий і логічний інтерфейс, що дозволяє швидко створювати вікна, кнопки, тексти, полі введення та інші елементи інтерфейсу. Tkinter також надає широкий набір вбудованих виджетів, які можна використовувати для створення різноманітних елементів інтерфейсу.

Tkinter підтримує інтеграцію з іншими бібліотеками та інструментами Python, що робить його універсальним і гнучким в розробці програм з графічним інтерфейсом. Він може бути використаний для розробки десктопних програм, інтерактивних дашбордів, ігор, інструментів аналізу даних та багатьох інших проєктів.

Також неможливо уявити розробку програмного забезпечення без використання системи контролю версій. Цей інструмент дозволяє розпаралелити розробку функцій між різними розробниками та забезпечує можливість відновлення кодової бази до певної версії. Вирішено використовувати систему контролю версій Git. Додатково, варто відзначити хмарну платформу для керування репозиторіями Bitbucket, надає можливість зберігання, спільного використання та керування версіями програмного коду. Одним з ключових функціональних можливостей Bitbucket є можливість створення приватних репозиторіїв, що забезпечує захист конфіденційної інформації і дозволяє командам розробників працювати в безпечному середовищі. Крім того, Bitbucket надає інструменти для керування правами доступу до репозиторіїв, що дозволяє контролювати, хто має право переглядати, редагувати або здійснювати злиття з кодовою базою.

Не менш важливим моментом є вибір IDE. Visual Studio Code (VSCode) з плагіном Jupyter виявився найбільш підходячим варіантом для написання

нейронної мережі. VSCode – це популярний текстовим редактором, який надає розширені можливості для розробки програмного забезпечення. За допомогою плагіна Jupyter, який встановлюється у VSCode, можна поєднати потужність редактора з можливостями Jupyter Notebook для розробки нейромереж. З використанням плагіна Jupyter у VSCode можна створювати та редагувати Jupyter-ноутбуки, які містять код нейромережі, а також виконувати їх прямо у редакторі. Плагін надає зручний інтерфейс для роботи з кодом, візуалізації результатів та взаємодії з нейромережами.

1.3.3 Аналіз відомих програмних продуктів

На сьогоднішній день існують програмні продукти для розпізнавання медичної маски на обличчі людини. Ось кілька з них:

– GetMaskCheck.com [11] - модель планшету компанії Apple iPad 8 із встановленим застосунком MaskCheck на спеціальному кріпленні. Розміщується дана система на вході у заклад, для отримання доступу до громадського закладу. Дану систему можна перепрофілювати як клієнтський кіоск з іншими застосунками (Рисунок 1.2).

Переваги:

– миттєві повідомлення про наявність або відсутність маски на особі людини;

– легко встановити та налаштувати застосунок на планшеті.

Недоліки:

– висока ціна за систему із-за додаткової покупки iPad 8;

– немає збору статистики за день;

– потребує постійне підключення до WI-FI мережі;

– відсутність конфіденційності, через закритий доступ до роботи програми є ризик несанкціонованого доступу (НСД) до інформації з боку розробника.

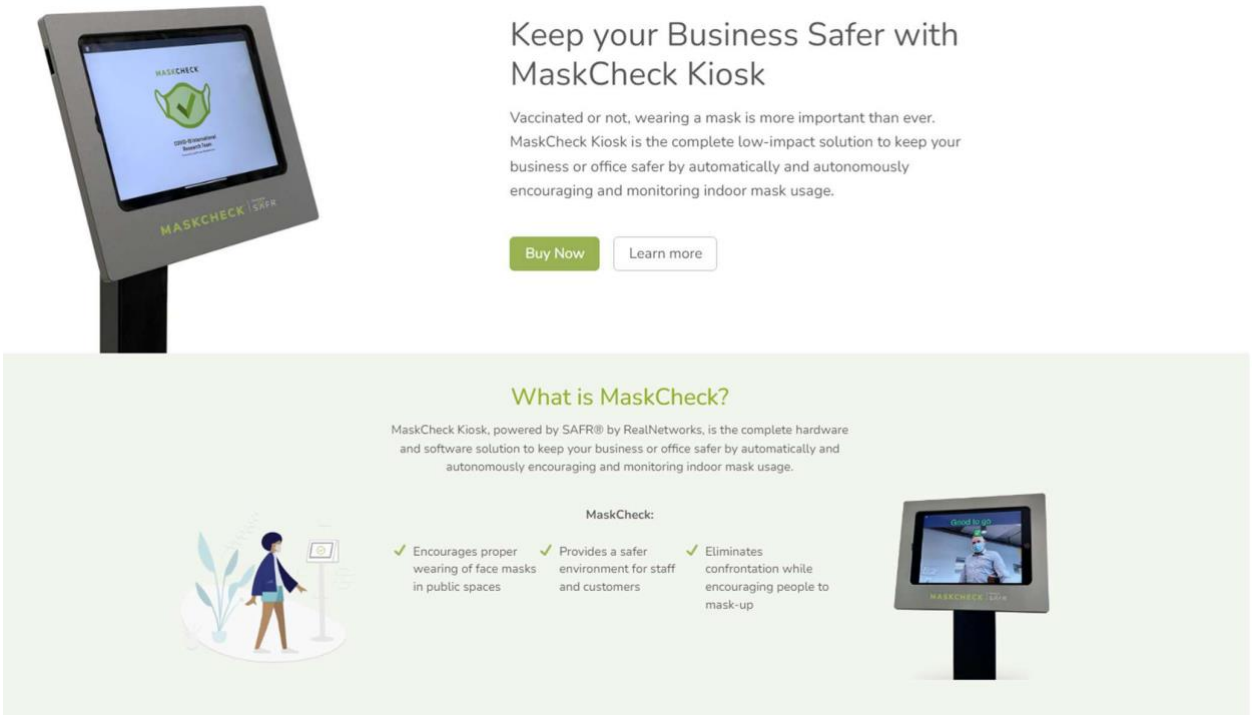

Рисунок 1.2 – Система розпізнавання MaskCheck

– SafeSpace[6] - це застосунок для захисту користувачів в закритих приміщеннях, який виявляє людей, що носять маски, за допомогою системи, заснованої на машинному навчанні. Цей додаток може працювати на будь-якому планшеті та був спеціально розроблений як рішення щодо безпеки в приміщенні для всіх видів бізнесу, включаючи роздрібну торгівлю, офісні будівлі, фабрики, входи до торгових центрів, ресторани та невеликі ринки.

Переваги:

– безкоштовний застосунок;

– надає налаштовані екранні повідомлення та звукові сповіщення;

– висока точність: 99,33% у виявленні масок для обличчя.

Недоліки:

– немає збору статистики;

– відсутність конфіденційності;

– тільки для Android та iOS пристроїв.

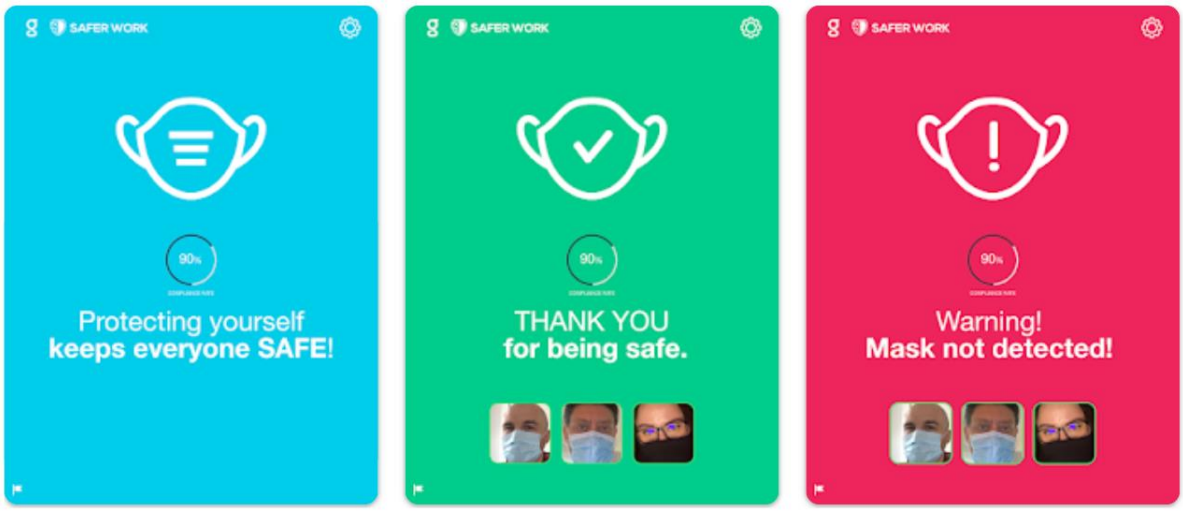

Рисунок 1.3 – Застосунок SafeSpace

Таблиця 1.1 відображає узагальнену порівняльну характеристику дипломного проєкту та аналогів

Таблиця 1.1 – Порівняльна характеристика наведених аналогів та дипломного проєкту

Функціонал	Дипломний проєкт	MaskCheck	SafeSpace
Визначення наявності маски	+	+	+
Необхідність підключення до мережі	-	+	+

Продовження таблиці 1.1

Можливість обробляти зображення в режимі реального часу	+	+	+
Збір статистики за день	+	-	-
Конфіденційність	-	-	-

1.4 Аналіз вимог до програмного забезпечення

Далі проведемо аналіз вимог до програмного забезпечення для виявлення аксесуарів на обличчі людини, а саме медичних масок. Цей етап має вагомий вплив на процес розробки програмного забезпечення, адже він визначає фундаментальні потреби та очікування від програмного забезпечення. Потрібно розуміти потреби користувача, щоб визначити функціональні та нефункціональні вимоги[13].

Функціональні вимоги визначають, які конкретні функції та операції повинно виконувати програмне забезпечення. Вони описують бажані результати та поведінку системи, яку користувачі очікують. Функціональні вимоги включають такі аспекти, як функції, операції, алгоритми, обмеження вхідних та вихідних даних, взаємодію з іншими системами та користувачами. Наприклад, функціональна вимога може вказувати, що система повинна забезпечувати можливість реєстрації користувачів, пошуку інформації або здійснення певних обчислень.

Нефункціональні вимоги визначають характеристики та якості програмного забезпечення, які не відносяться безпосередньо до його

функціональності. Вони описують вимоги до продуктивності, надійності, безпеки, масштабованості, зручності використання, сумісності з іншими системами, а також вимоги до документації, підтримки та інші некоректні аспекти. Наприклад, нефункціональна вимога може стосуватися швидкодії системи, що вимагає, щоб вона обробляла дані у певний строк, або вимоги до безпеки, що визначають, які механізми безпеки мають бути впроваджені для захисту від несанкціонованого доступу.

Діаграма варіантів використання програмного продукту наведена у графічних матеріалах «Схема структурна варіантів використання».

Більш детальний опис кожного варіанту використання наведений в таблицях 1.2-1.7.

Таблиця 1.2 – Варіант використання UC-01

Назва	Авторизація
Ідентифікатор	UC-01
Актор	Неавторизований користувач
Передумови	Користувач має профіль в застосунку
Основний сценарій	Користувач переходить на сторінку авторизації, вводить логін та пароль, натискає на кнопку «Увійти».
Постумови	Користувач авторизується в системі та перенаправляється на головну сторінку.
Розширений сценарій	Повідомлення про те, що деякі поля на формі були введені некоректно

Таблиця 1.3 – Варіант використання UC-02

Назва	Реєстрація користувача
Ідентифікатор	UC-02
Актор	Незареєстрований користувач
Передумови	У користувача немає профіля в застосунку

Продовження таблиці 1.3

Основний сценарій	Користувач переходить на сторінку реєстрації, вводить логін та пароль та натискає на кнопку «Реєстрація»
Постумови	Створення сторінки користувача
Розширений сценарій	Повідомлення про те, що деякі поля на формі були введені некоректно.

Таблиця 1.4 – Варіант використання UC-03

Назва	Перегляд зображення з камери в режимі реального часу
Ідентифікатор	UC-03
Актор	Авторизований користувач
Передумови	Користувач має профіль в застосунку
Основний сценарій	Користувач запускає процес обробки відео з камери
Постумови	Користувач переглядає відео з камери в режимі реального часу
Розширений сценарій	Оброблені зображення зберігаються в базу даних

Таблиця 1.5 – Варіант використання UC-04

Назва	Переглянути статистику
Ідентифікатор	UC-04
Актор	Авторизований користувач
Передумови	Користувач має профіль в застосунку
Основний сценарій	Користувач натискає на кнопку «Статистика»
Постумови	Перегляд статистики наявності/відсутності масок

Продовження таблиці 1.5

Розширений сценарій	Перегляд конкретних збережених зображень

Таблиця 1.6 – Варіант використання UC-05

Назва	Переглянути користувачів
Ідентифікатор	UC-05
Актор	Адміністратор
Передумови	Користувач є адміністратором
Основний сценарій	Адміністратор натискає на кнопку «Переглянути користувачів»
Постумови	Перегляд списку користувачів
Розширений сценарій	-

Таблиця 1.7 – Варіант використання UC-06

Назва	Видалити користувача
Ідентифікатор	UC-06
Актор	Адміністратор
Передумови	Користувач є адміністратором та знаходить на сторінці перегляду списку користувачів
Основний сценарій	Адміністратор натискає на кнопку конкретного користувача та натискає кнопку «Видалити»
Постумови	Користувача видалено з бази даних, з’являється повідомлення про успіх дії
Розширений сценарій	-

1.4.1 Розроблення функціональних вимог

Модель вимог, створена на основі детального аналізу варіантів використання продукту, описана у таблиці 1.8:

Таблиця 1.8 – Модель вимог

Вимога	Код	Пріоритет	Ризик
1. Реєстрація користувача	FR-01	Високий	Середній
1.1 Валідація полів	FR-01.1	Низький	Низький
1.2 Перевірка існування користувача з введеним логіном	FR-01.2	Високий	Низький
2.Авторизація користувача	FR-02	Високий	Середній
2.1 Валідація полів	FR-02.1	Низький	Низький
3. Перегляд зображень	FR-03	Високий	Середній
3.1 Перегляд зображень людей без масок	FR-03.1	Високий	Середній
3.2 Перегляд зображень людей з масками	FR-03.2	Високий	Середній
4. Перегляд статистики	FR-04	Високий	Середній
4.1 Перегляд статистики по даті	FR-04.1	Високий	Середній
4.2 Перегляд статистики по наявності маски	FR-04.2	Високий	Середній
5. Перегляд зображення з камери в режимі реального часу	FR-05	Високий	Середній
6. Перегляд списку користувачів	FR-06	Високий	Високий
7. Видалення користувача	FR-07	Високий	Високий

В таблицях 1.9-1.15 наведений опис основних функціональних вимог.

Таблиця 1.9 – Функціональна вимога FR-01

Ідентифікатор	FR-01
Назва	Реєстрація користувача
Опис	Застосунок надає можливість користувачеві зареєструватись

Таблиця 1.10 – Функціональна вимога FR-02

Ідентифікатор	FR-02
Назва	Авторизація користувача
Опис	Застосунок надає можливість користувачеві авторизуватись

Таблиця 1.11 – Функціональна вимога FR-03

Ідентифікатор	FR-03
Назва	Перегляд зображень
Опис	Застосунок надає можливість переглянути зображення

Таблиця 1.12 – Функціональна вимога FR-04

Ідентифікатор	FR-04
Назва	Перегляд статистики
Опис	Застосунок надає можливість користувачеві переглянути статистику

Таблиця 1.13 – Функціональна вимога FR-05

Ідентифікатор	FR-05

Продовження таблиці 1.13

Назва	Перегляд зображення з камери в режимі реального часу
Опис	Застосунок надає можливість користувачеві переглянути зображення з камери в режимі реального часу

Таблиця 1.14 – Функціональна вимога FR-06

Ідентифікатор	FR-06
Назва	Перегляд списку користувачів
Опис	Застосунок надає можливість адміністратору переглянути список всіх користувачів

Таблиця 1.15 – Функціональна вимога FR-07

Ідентифікатор	FR-07
Назва	Видалення користувача
Опис	Застосунок надає можливість адміністратору видалити певного користувача

Проведений аналіз варіантів використання та функціональних вимог дозволяє встановити відповідність між ними, що зображено в матриці трасування 1.16:

Таблиця 1.16 – Матриця трасування

	FR-01	FR-02	FR-03	FR-04	FR-05	FR-06	FR-07
UC-01		x					
UC-02	x						

Продовження таблиці 1.16

UC-03					x		
UC-04			x	x			
UC-05						x	
UC-06							x

1.4.2 Розроблення нефункціональних вимог

Під час аналізу програмного продукту виділено наступні нефункціональні вимоги:

– Точність розпізнавання: Головною вимогою є висока точність розпізнавання медичної маски. Програмне забезпечення повинно здати впевнено визначати, чи присутня медична маска на обличчі або ні, мінімізуючи помилки.

– Швидкодія: Вимога до швидкодії залежить від конкретного використання програмного забезпечення. Деякі випадки можуть вимагати миттєвої відповіді, особливо в області безпеки або медицини, де час має критичне значення.

– Масштабованість: Якщо програмне забезпечення планується використовувати на великій кількості об'єктів або в розподіленому середовищі, важливо мати можливість масштабування ресурсів та продуктивності для забезпечення ефективної роботи.

– Адаптація до змінних умов: При розпізнаванні медичних масок можуть виникати різні умови, такі як різні освітлення, пози обличчя, фони тощо. Вимога полягає у високій стійкості до змінних умов для надійного розпізнавання.

– Підтримка різних платформ: Залежно від конкретного застосування може знадобитися підтримка різних платформ, таких як комп'ютери, мобільні пристрої, вбудовані системи. Вимога полягає у

можливості розгортання та роботі програмного забезпечення на різних пристроях.

– Простота інтеграції: Якщо програмне забезпечення має інтегруватися з іншими системами або службами, вимога полягає у простоті інтеграції та можливості обміну даними з іншими компонентами системи.

– Безпека і конфіденційність: У випадку розпізнавання медичних масок можуть бути важливі питання безпеки та конфіденційності даних. Вимога полягає у забезпеченні високого рівня захисту персональних даних та забезпеченні конфіденційності користувачів.

– Легкість використання та налаштування: Інтерфейс користувача програмного забезпечення повинен бути зрозумілим та легким у використанні, а процес налаштування мережі масок повинен бути простим та інтуїтивно зрозумілим.

– Навчання та підтримка: Деякі користувачі можуть потребувати навчання або підтримки при використанні програмного забезпечення. Вимога полягає у наявності документації, навчальних матеріалів та підтримки від розробника.

– Вартість та ліцензування: Вимоги до вартості та ліцензування програмного забезпечення можуть різнитися в залежності від бюджету та потреб користувача. Важливо враховувати доступні фінансові можливості та моделі ліцензування програмного забезпечення.

1.5 Постановка задачі

Отже, в рамках даного дипломного проєкту планується розробити програмне забезпечення, яке може визначати наявність аксесуарів на обличчі людини, у тому числі медичної маски.

Метою даного проєкту є покращення безпеки, за допомогою відстеження в медичних та державних установах визначення наявності аксесуарів на обличчі людини на основі нейронної мережі.

Головними функціональними задачами є розпізнавання наявності аксесуарів на обличчі людини, надання користувачеві статистики та можливість переглядати збережені фото.

Висновки до розділу

В цьому розділі проведено глибокий аналіз предметної області, використовуючи широкий спектр джерел, таких як наукові статті та статистичні дані. Цей етап дослідження дозволив виявити основні проблеми, що характерні для цієї області, а також знайти потенційні шляхи їх вирішення.

Після цього проаналізовано наявне на сьогодні алгоритмічне забезпечення та технологічні рішення, що вже застосовуються у цій області, з метою визначити ті з них, які можуть допомогти в реалізації програмного забезпечення.

Наступним пунктом здійснено детальний огляд наявних програмних засобів та середовищ розробки. Це дослідження допомогло визначити оптимальні рішення щодо використання засобів розробки та допоміжних програмних засобів.

Далі проведено огляд існуючих програмних продуктів, що пов'язані з даною областю, та ретельно проаналізовано їх переваги та недоліки. Також здійснено порівняння функціональності цих продуктів з функціональністю програмного забезпечення, яке буде розроблятись в рамках дипломного проєкту.

Наступним кроком розглянуто різноманітні варіанти використання програмного забезпечення і сформульовано функціональні та нефункціональні вимоги.

У кінцевому результаті були чітко сформульовані конкретні задачі, які необхідно вирішити під час розробки програмного забезпечення.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Моделювання та аналіз програмного забезпечення

Моделювання програмного забезпечення є важливим етапом у процесі розробки, оскільки воно дозволяє уявити, створити і протестувати систему до її фактичної реалізації. Це дає можливість виявити та виправити потенційні проблеми, зрозуміти вимоги користувачів та визначити оптимальну архітектуру системи. Моделі програмного забезпечення допомагають визначити логіку роботи системи, структуру даних, взаємозв'язки між компонентами та потрібні функціональність. Вони також допомагають комунікувати зі зацікавленими сторонами та забезпечують базу для подальшої розробки, тестування та підтримки програмного забезпечення. В результаті моделювання програмного забезпечення дозволяє покращити якість та ефективність розробки, зменшити ризики і вартість впровадження системи, а також покращити задоволення користувачів та виконання їх вимог.

BPMN дозволяє моделювати послідовність дій, паралельність, умови, цикли та інші аспекти бізнес-процесів. Вона дозволяє розуміти структуру та логіку процесу, ідентифікувати його етапи, взаємозв'язки та залежності. Крім того, BPMN може бути використана для аналізу, оптимізації та автоматизації бізнес-процесів.

Завдяки своїй стандартизованій нотації, BPMN дозволяє різним стейкхолдерам, таким як бізнес-аналітики, розробники та менеджери, легко спілкуватися та співпрацювати під час розробки, вдосконалення та автоматизації бізнес-процесів.

Далі, на Рисунках 2.1-2.3 наведено змодельовані процесу продукту.

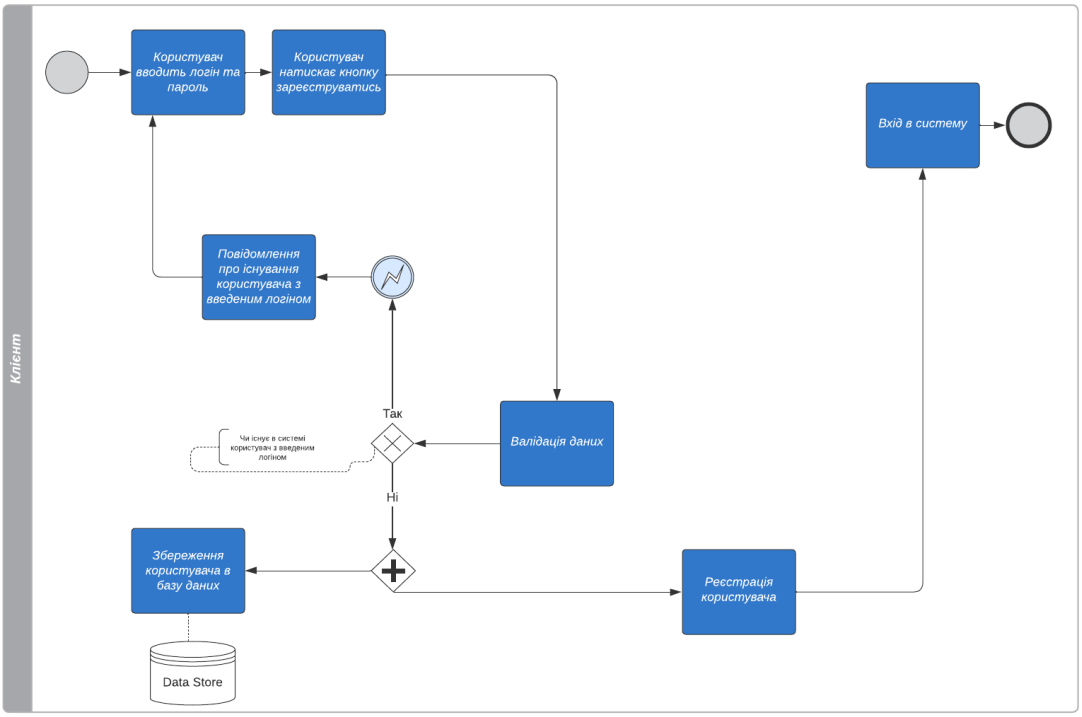

Рисунок 2.1 – Реєстрація користувача

Послідовний опис процесу реєстрації користувача:

– користувачу відображається сторінка реєстрації;

– користувач заповнює всі необхідні поля для реєстрації;

– перевіряються дані користувача на валідність, шифруються та додають в базу даних в таблицю «User»;

– якщо введені поля неправильно заповнені, то видається помилка.

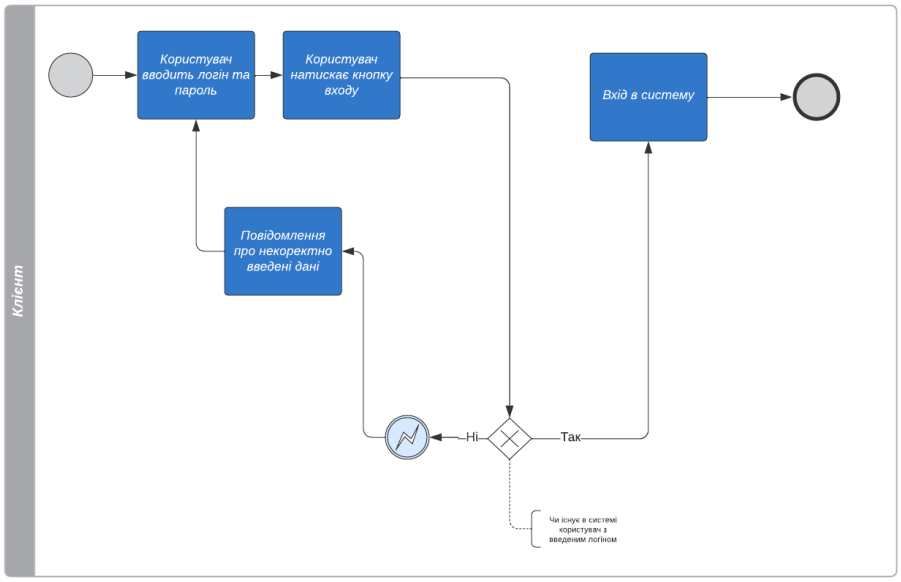

Рисунок 2.2 – Авторизація користувача

Послідовний опис процесу авторизації користувача:

– користувачу відображається сторінка авторизації;

– користувач заповнює всі необхідні поля для авторизації;

– перевіряються дані користувача на валідність;

– якщо введені поля неправильно заповнені, то видається помилка.

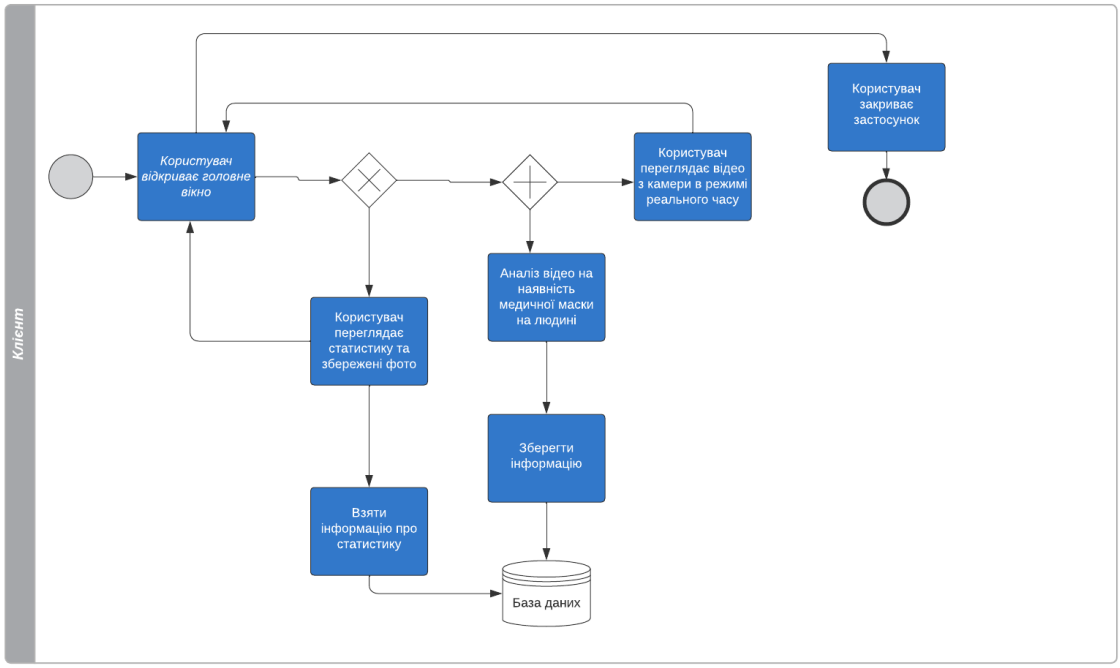

Рисунок 2.3 – Інші процеси

Послідовний опис процесу авторизації користувача:

– користувачу відображається головна сторінка;

– користувач переходить до перегляду статистики;

– з бази даних береться інформація та відображається користувачу;

– користувач переходить до аналізу відео на наявність медичної маски;

– отримана інформація зберігається в базу даних в таблицю “File”;

– користувач виходить із застосунку.

Діаграма послідовностей - це візуальне зображення послідовності кроків або етапів, необхідних для виконання певного процесу. Вона використовується для ясного представлення послідовності дій та

взаємозв'язків між ними. Діаграма процесів допомагає краще розуміти і візуалізувати процес, розбиваючи його на більш зрозумілі та логічні кроки.

Діаграма послідовностей наведена у графічних матеріалах «Схема структурна послідовностей».

На цій діаграмі зображено 3 процеси, а саме: авторизацію, запуск обробки відео з камери в режимі реального часу та відображення обробленого відео та підрахунок і відображення статистики.

2.2 Архітектура програмного забезпечення

Так як головною складовою даного застосунку є нейромережа, то його архітектура виглядає наступним чином (Рисунок 2.4)

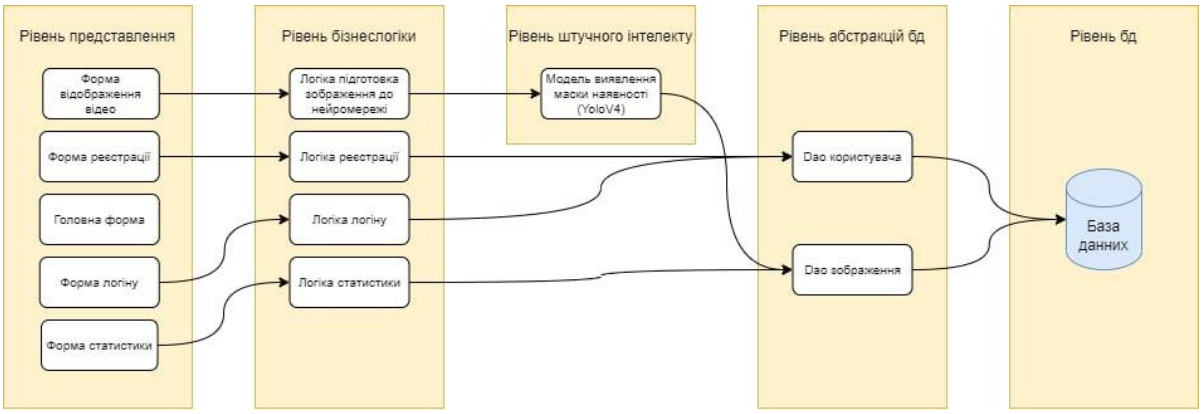

Рисунок 2.4 – Архітектура програмного забезпечення.

Рівень представлення включає в себе графічний інтерфейс користувача, який складається з форми реєстрації, форми логіну, головної форми, форми відображення відео та форми статистики.

Рівень бізнес логіки відповідає за обробку та управління бізнес-процесами та логікою додатку. Цей рівень забезпечує виконання основних функцій та завдань, а саме логіку авторизації, реєстрації, підготовки зображення до нейромережі, перегляду статистики.

Оскільки, в даному програмного забезпеченні акцентується увага на нейромережі, то під неї було виділено окремий рівень архітектури, до якого має доступ рівень бізнес логіки.

Рівень абстракції БД відповідає за взаємодію між додатком і самою базою даних та складається з двох сутностей: користувача та зображення.

Рівень бази даних (БД) відповідає за фізичне зберігання та керування даними. Його основна мета - забезпечити ефективну та надійну роботу з даними у системі.

Найлегше зобразити роботу даного програмного забезпечення з допомогою flow діаграми на Рисунку 2.5.

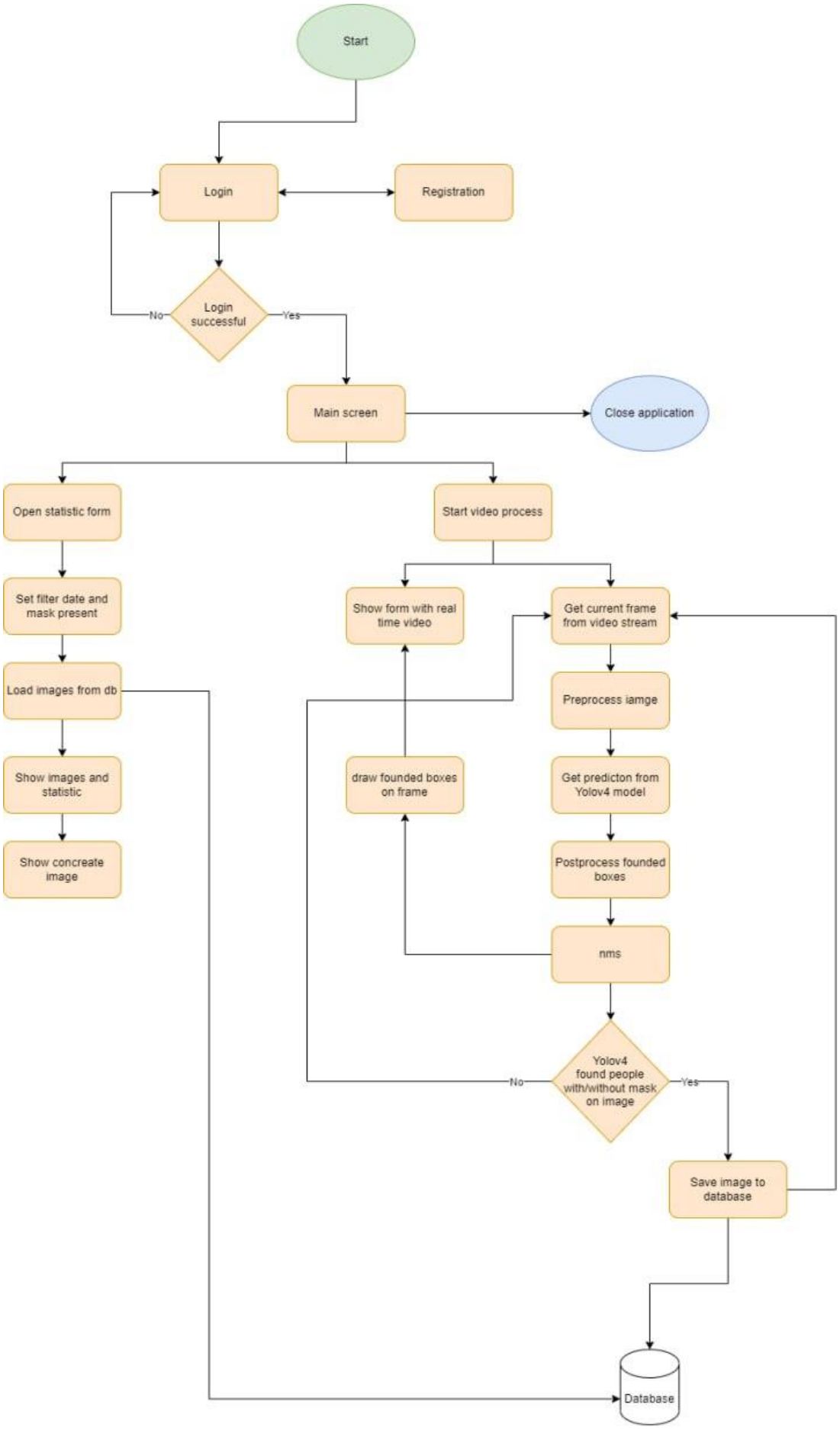

Рисунок 2.5 – Flow діаграма програмного забезпечення

Обов’язково треба згадати про архітектуру нейронної мережі YOLOv4. Її будова зображена на Рисунку 2.6

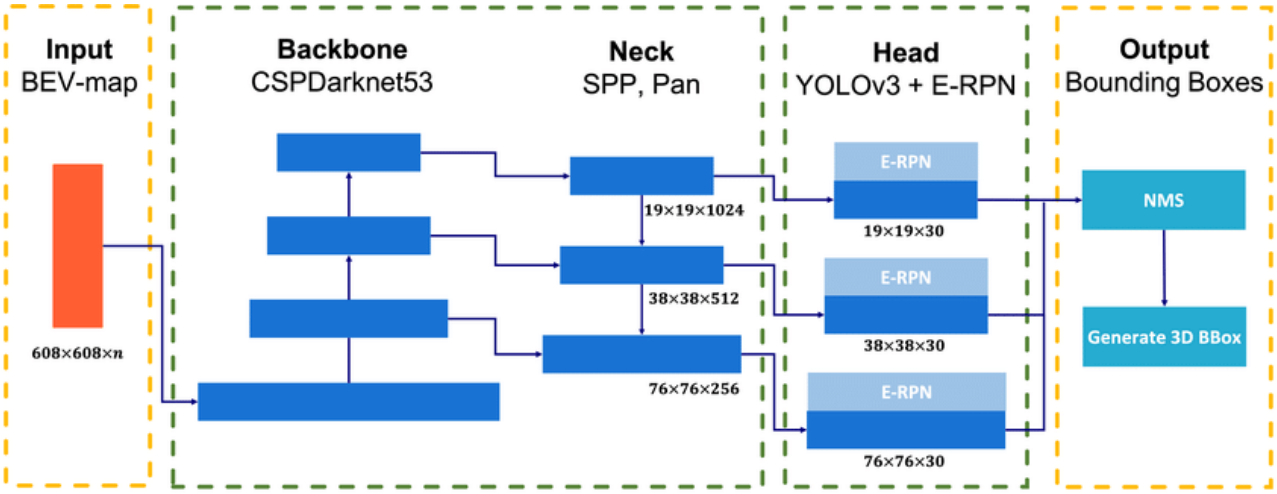

Рисунок 2.6 – Архітектура нейронної мережі YOLOv4

YOLOv4 (You Only Look Once version 4) - це одна з найновіших версій YOLO, системи розпізнавання об'єктів в реальному часі. YOLOv4 включає ряд важливих оновлень, які спрямовані на покращення швидкості та точності.

– Backbone: Основа архітектури YOLOv4 - це CSPDarknet53. Це модифікація Darknet53, яка включає Cross Stage Partial connections (CSP). Ця структура дозволяє моделі бути глибшою, не втрачаючи при цьому швидкості.

– Neck: Шия в YOLOv4 використовує PANet та SAM block. PANet дозволяє моделі ефективно використовувати признаки з різних рівнів, а SAM block допомагає зосередитись на важливих областях зображення.

– Head: Голова в YOLOv4 використовує YOLOv3 head з трьома різними розмірами якорів для кожного з трьох рівнів виходу. Це дозволяє моделі виявляти об'єкти різних розмірів.

– Методи збільшення точності: YOLOv4 включає ряд методів для покращення точності, включаючи Mish activation, CIOU loss, и Cross mini-Batch Normalization.

– Методи збільшення швидкості: YOLOv4 також включає методи для покращення швидкості, включаючи CIOU loss, Eliminate grid sensitivity,

Using multiple anchors for a single ground truth, Cosine annealing scheduler, Optimal hyperparameters, Random training shapes, та інші.

– Методи збільшення стабільності: Для покращення стабільності YOLOv4 використовує Normalizer-Free Training, Cross-Stage Partial Connections (CSP), Multi-input Weighted Residual Connections (MiWRC).

Всі ці елементи разом дають YOLOv4 високу швидкість та точність виявлення об'єктів.

2.3 Конструювання програмного забезпечення

Розробку нейромережі варто почати з підготовки датасету. Розробка датасету для навчання нейронної мережі є важливим етапом, який включає збір та розмітку даних. Даний датасет складається з зображень людей в масках та без масок, які були зібрані з Інтернету. Давайте розглянемо цей процес детальніше:

– Збір зображень: Було зібрано зображення людей в масках та без масок з Інтернету. Це може включати в себе пошук зображень за допомогою пошукових систем, завантаження з відкритих датасетів або використання API для збору зображень з соціальних медіа або інших веб-сайтів. У конкретно даному випадку було взято відкритий датасет.

– Розмітка зображень: Після збору зображень використовується програма Labellmg для розмітки датасету. LabelImg - це графічний інструмент розмітки зображень, який дозволяє вручну вказувати області (або "бокси") на зображеннях, де знаходяться об'єкти, які користувач хоче, щоб нейронна мережа виявляла. Вказується, де на зображенні знаходиться людина, і чи має вона на собі маску. (Рисунок 2.7)

Рисунок 2.7 – Розмітка зображень

– Класи датасету: Датасет має два класи: люди з масками та люди без масок. Класи - це категорії, до яких можуть належати об'єкти на зображеннях. Ці класи використовуються при розмітці зображень, щоб вказати, до якого класу належить кожен об'єкт.

Після того, як зібрали та розмітили датасет, ви можна використовувати його для тренування вашої нейронної мережі. Мережа буде вчитися визначати, чи має людина на зображенні маску, на основі прикладів, які надано в датасеті.

Після збору та розмітки зображень, потрібно перетворити дані в формат, який придатний для YOLOv4 (Рисунок 2.8 та 2.9).

```python
#Converting data at sight necessary for training Yolov4
def parse_xml(img_folder, file):
    for xml_file in glob.glob(img_folder+'/*.xml'):
        tree=ET.parse(open(xml_file))
        root = tree.getroot()
        image_name = root.find('filename').text
        img_path = img_folder+'/'+image_name
        for i, obj in enumerate(root.iter('object')):
            # difficult = obj.find('difficult').text
            cls = obj.find('name').text
            if cls not in Dataset_names:
                Dataset_names.append(cls)
            cls_id = Dataset_names.index(cls)
            xmlbox = obj.find('bndbox')
            OBJECT = (str(int(float(xmlbox.find('xmin').text)))+','
                      +str(int(float(xmlbox.find('ymin').text)))+','
                      +str(int(float(xmlbox.find('xmax').text)))+','
                      +str(int(float(xmlbox.find('ymax').text)))+','
                      +str(cls_id))
            img_path += ' '+OBJECT
        img_path = img_path.replace("\\", "/")
        img_path = img_path.replace("pandemic masks", "pandemic_masks")
        print(img_path)
        file.write(img_path+'\n')
```

```python
#Converting data at sight necessary for training Yolov4
def convert_data_for_yolov4():
    for i, folder in enumerate(['train','test']):
        with open([TRAIN_ANNOT_PATH,TEST_ANNOT_PATH][i], "w") as file:
            print(os.getcwd()+DATA_DIR+folder)
            img_path = os.path.join(os.getcwd()+DATA_DIR+folder)
            for directory in os.listdir(img_path):
                xml_path = os.path.join(img_path, directory)
                parse_xml(xml_path, file)

    print("Dataset_names:", Dataset_names)
    with open(DATASET_NAMES_CLASSES, "w") as file:
        for name in Dataset_names:
            file.write(str(name)+'\n')
```

Рисунок 2.8 – Конвертація даних в формат YOLOv4

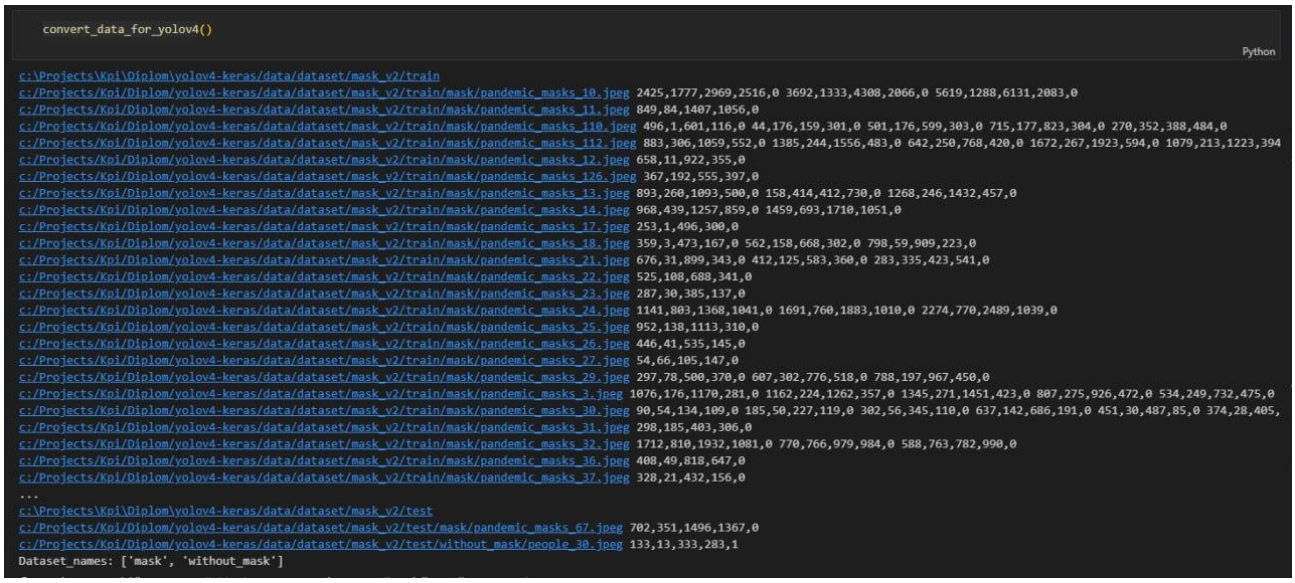

Рисунок 2.9 – Конвертація даних в формат YOLOv4

Після перетворення даних в формат, придатний для YOLOv4, створюється кастомний клас для завантаження цих зображень та міток.

Розглянемо наступні етапи:

– Ініціалізація функцій втрат. Ініціалізуються функції втрат IOU (Intersection Over Union) та GIOU (Generalized Intersection Over Union). IOU - це метрика, яка вимірює перетин двох областей (наприклад, обмежувального прямокутника об'єкта та прогнозованого обмежувального прямокутника). GIOU - це розширення IOU, яке також враховує геометричну форму обмежувальних прямокутників.

– Вказівка гіперпараметрів навчання. Гіперпараметри - це параметри, які встановлюються до початку процесу навчання і не змінюються під час навчання. Встановлюється розмір пакету (BATCH_SIZE) рівним 4, початкову швидкість навчання (TRAIN_LR_INIT) рівну 1e-3, кінцеву швидкість навчання (TRAIN_LR_END) рівну 1e-6, та кількість епох (EPOCHS) рівну 100.

– Вибір оптимізатора. Використовується оптимізатор Adam для мінімізації функції втрат. Adam - це алгоритм оптимізації, який використовує методи адаптивного оцінювання моментів. Він вважається ефективним для навчання глибоких нейронних мереж, оскільки він автоматично налаштовує

швидкість навчання на основі оцінки першого та другого моментів градієнтів. В порівнянні з іншими оптимізаторами, такими як SGD (Stochastic Gradient Descent) або RMSprop, Adam часто пропонує кращу швидкість збіжності та стабільність.

На Рисунку 2.10 наведено приклад логів тренування.

```
epoch:34 step:   14/25, lr:0.000752, giou_loss:   2.38, conf_loss:   2.54, prob_loss:   3.34, total_loss:   8.26
epoch:34 step:   15/25, lr:0.000751, giou_loss:   3.64, conf_loss:   3.54, prob_loss:   5.86, total_loss:  13.05
epoch:34 step:   16/25, lr:0.000751, giou_loss:   4.83, conf_loss:   4.70, prob_loss:   4.09, total_loss:  13.62
epoch:34 step:   17/25, lr:0.000750, giou_loss:   7.29, conf_loss:   5.93, prob_loss:   5.75, total_loss:  18.96
epoch:34 step:   18/25, lr:0.000750, giou_loss:   3.04, conf_loss:   3.11, prob_loss:   2.78, total_loss:   8.92
epoch:34 step:   19/25, lr:0.000749, giou_loss:   2.11, conf_loss:   2.66, prob_loss:   2.26, total_loss:   7.03
epoch:34 step:   20/25, lr:0.000748, giou_loss:   2.68, conf_loss:   3.17, prob_loss:   3.18, total_loss:   9.04
epoch:34 step:   21/25, lr:0.000748, giou_loss:   0.99, conf_loss:   1.81, prob_loss:   1.27, total_loss:   4.07
epoch:34 step:   22/25, lr:0.000747, giou_loss:   2.14, conf_loss:   3.12, prob_loss:   1.38, total_loss:   6.64
epoch:34 step:   23/25, lr:0.000747, giou_loss:   3.50, conf_loss:   4.19, prob_loss:   2.05, total_loss:   9.74
epoch:34 step:   24/25, lr:0.000746, giou_loss:   1.09, conf_loss:   2.04, prob_loss:   0.93, total_loss:   4.06
epoch:34 step:    0/25, lr:0.000746, giou_loss:   1.41, conf_loss:   2.08, prob_loss:   2.49, total_loss:   5.98
epoch:34 step:    1/25, lr:0.000745, giou_loss:   2.31, conf_loss:   2.42, prob_loss:   3.02, total_loss:   7.75


giou_val_loss:   1.40, conf_val_loss:   3.64, prob_val_loss:   6.14, total_val_loss:  11.18


epoch:35 step:    2/25, lr:0.000744, giou_loss:   1.28, conf_loss:   2.09, prob_loss:   2.74, total_loss:   6.11
epoch:35 step:    3/25, lr:0.000744, giou_loss:   2.91, conf_loss:   3.91, prob_loss:   4.04, total_loss:  10.86
epoch:35 step:    4/25, lr:0.000743, giou_loss:   2.78, conf_loss:   2.52, prob_loss:   4.14, total_loss:   9.45
epoch:35 step:    5/25, lr:0.000743, giou_loss:   3.16, conf_loss:   3.65, prob_loss:   5.16, total_loss:  11.96
epoch:35 step:    6/25, lr:0.000742, giou_loss:   3.30, conf_loss:   3.32, prob_loss:   4.24, total_loss:  10.86
epoch:35 step:    7/25, lr:0.000742, giou_loss:   8.04, conf_loss:   8.11, prob_loss:   5.19, total_loss:  21.33
epoch:35 step:    8/25, lr:0.000741, giou_loss:   3.30, conf_loss:   3.20, prob_loss:   2.72, total_loss:   9.21
epoch:35 step:    9/25, lr:0.000741, giou_loss:   3.89, conf_loss:   3.94, prob_loss:   6.54, total_loss:  14.37
epoch:35 step:   10/25, lr:0.000740, giou_loss:   2.32, conf_loss:   2.72, prob_loss:   1.79, total_loss:   6.82
epoch:35 step:   11/25, lr:0.000739, giou_loss:   1.13, conf_loss:   1.74, prob_loss:   1.85, total_loss:   4.72
epoch:35 step:   12/25, lr:0.000739, giou_loss:   3.18, conf_loss:   3.95, prob_loss:   2.47, total_loss:   9.60
```

Рисунок 2.10 – Логи тренування

На Рисунку 2.11 зображено діаграму БД.

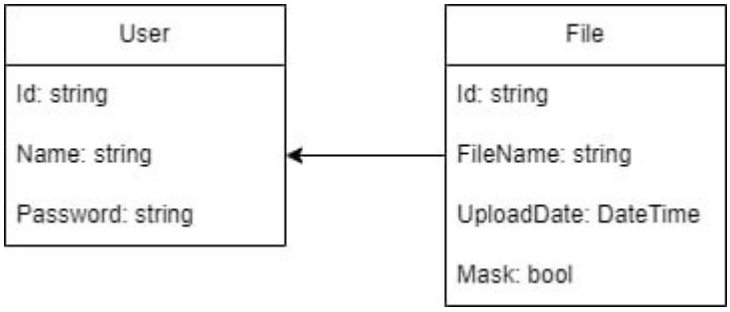

Рисунок 2.11

У таблицях 2.1 та 2.2 наведено опис основних таблиць та атрибутів бази даних

Таблиця 2.1 – Опис таблиць бази даних

Назва	Опис
User	Зберігається інформація про користувача
File	Зберігається інформація про файл

Таблиця 2.2 - Опис атрибутів бази даних

Таблиця	Атрибут	Тип	Опис
User	Id	String	Ідентифікатор
	Name	String	Ім’я
	Password	String	Пароль
File	Id	String	Ідентифікатор
	Filename	String	Назва файлу
	UploadDate	DateTime	Дата загрузки
	Mask	Bool	Чи наявна маска

2.4 Аналіз безпеки даних

Аналіз безпеки даних є ключовим аспектом при створенні програмного забезпечення. Цей процес включає в себе оцінку захисту персональних даних користувачів, а також виявлення та усунення потенційних вразливостей програмного забезпечення.

GDPR (General Data Protection Regulation) є регуляторним законодавством, що вступило в силу в Європейському Союзі у 2018 році з метою захисту приватності та обробки персональних даних громадян. Цей регламент визначає правила та обов'язки організацій, що збирають, зберігають, обробляють та передають персональні дані.

GDPR надає громадянам більший контроль над їх персональними даними, вимагаючи згоди на збір та обробку цих даних, а також забезпечує їх право на доступ до своїх даних, виправлення неточностей, видалення та

перенесення даних. Він також накладає вимоги щодо безпеки та конфіденційності персональних даних, включаючи обов'язок повідомляти про порушення безпеки даних.

Також варто згадати SQL-ін'єкцію - це тип атаки на безпеку, при якому зловмисник вставляє шкідливий код SQL в запит до бази даних через веб-інтерфейс. Це може призвести до несанкціонованого доступу до даних, їх зміни або видалення.

Є кілька способів захисту від SQL-ін'єкцій:

– Використання параметризованих запитів або запитів з підстановкою: Це означає, що потрібно вказувати структуру SQL-запиту, а потім надавати кожному параметру конкретне значення. Це забезпечує, що введені користувачем дані не можуть змінити структуру запиту.

– Використання ORM-бібліотек: Об'єктно-реляційні відображення (ORM) - це техніка програмування, яка перетворює дані між системами типів, що несумісні на перший погляд. Використання ORM може допомогти запобігти SQL-ін'єкціям.

– Валідація та санітарна обробка вхідних даних: Для цього треба переконатися, що вхідні дані відповідають очікуваному формату та типу. Наприклад, якщо очікується числове значення, треба відхилити будь-які вхідні дані, які містять літери або спеціальні символи.

– Обмеження привілеїв: Потрібно надати базам даних та користувачам найменше необхідне рівень привілеїв. Це зменшує потенційний збиток від атаки SQL-ін'єкції.

– Регулярне оновлення та патчування СУБД: Виробники баз даних регулярно випускають оновлення та патчі, які виправляють відомі проблеми безпеки.

Висновки до розділу

У цьому розділі виявлено ключові бізнес-процеси, що пов'язані з розроблюваним продуктом, і проведений їх детальний аналіз. Цей аналіз дозволив розібратися в основних компонентах програмного забезпечення і з'ясувати, як вони взаємодіють між собою. Результатом цього аналізу стала чітка уява про структуру продукту та відношення між його складовими, що стало фундаментом для подальшого проєктування і розробки.

На наступному етапі була проведена глибока аналіз архітектури системи, зокрема архітектури нейронної мережі. Вивчення архітектурних принципів дозволило визначити оптимальну структуру системи та взаємозв'язки між її компонентами. Були враховані функціональні та нефункціональні вимоги до системи, а також здійснено аналіз можливих ризиків та вибрані підходи до забезпечення швидкодії, масштабованості та стабільності архітектури.

Після завершення проєктування архітектури настала фаза реалізації системи, яка включала розробку нейронної мережі та інших необхідних компонентів програмного забезпечення. Крок за кроком проведено процес підготовки даних, включаючи збір, очищення і форматування даних, а також їх розподіл на навчальний та перевірочний набори. Далі відбулося навчання нейронної мережі з використанням навчальних даних, з метою досягнення високої ефективності та точності системи.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Аналіз якості програмного забезпечення є важливим етапом в розробці програмних продуктів. Він забезпечує оцінку якості коду та виявлення різних проблем, що можуть виникнути під час роботи програми. Для проведення аналізу використовуються спеціальні інструменти, які можуть виконувати статичний та динамічний аналіз коду.

Статичний аналіз коду здійснюється без фактичного виконання програми. Його основна мета - перевірити кодову базу на наявність помилок, проблем стилю коду, потенційних вразливостей та інших недоліків. Під час статичного аналізу використовуються різні алгоритми та правила, які дозволяють виявляти проблемні місця в коді. Наприклад, можна перевіряти дублювання коду, неправильність використання змінних, проблеми з пам'яттю та інші типові помилки.

Динамічний аналіз коду вимагає фактичного виконання програми або його окремих частин. Під час динамічного аналізу оцінюються різні аспекти функціонування програми, такі як безпека, продуктивність, використання пам'яті та покриття коду тестами. Цей аналіз може здійснюватися шляхом спеціальних інструментів для відстеження викликів функцій, використання ресурсів, виявлення можливих помилок або потенційних проблем.

У результаті статичного та динамічного аналізу отримується інформація про якість програмного забезпечення, виявлені проблеми та рекомендації щодо їх виправлення. Це дозволяє розробникам покращити якість свого коду, забезпечити безпеку програмного забезпечення та підвищити ефективність його роботи.

Для оцінки якості програмного забезпечення вирішено використовувати SonarQube. Це потужний інструмент, який допомагає виконувати аналіз, забезпечуючи широкий спектр функціональностей. Він

проводить статичний аналіз для перевірки кодової бази на відповідність стандартам програмування та виявлення проблем. Крім того, SonarQube виконує динамічний аналіз, оцінюючи функціонування програмного забезпечення в реальному часі. Він надає метрики та звіти, які допомагають розробникам аналізувати результати аналізу та приймати необхідні кроки для поліпшення якості програми. З його допомогою можна легко інтегрувати аналіз коду в проєкти розробки та забезпечити постійний контроль якості. SonarQube є популярним інструментом серед розробників, допомагаючи покращити якість програмного забезпечення та забезпечити безпеку коду.

На Рисунку 3.1 представлений звіт SonarQube після аналізу програмного забезпечення.

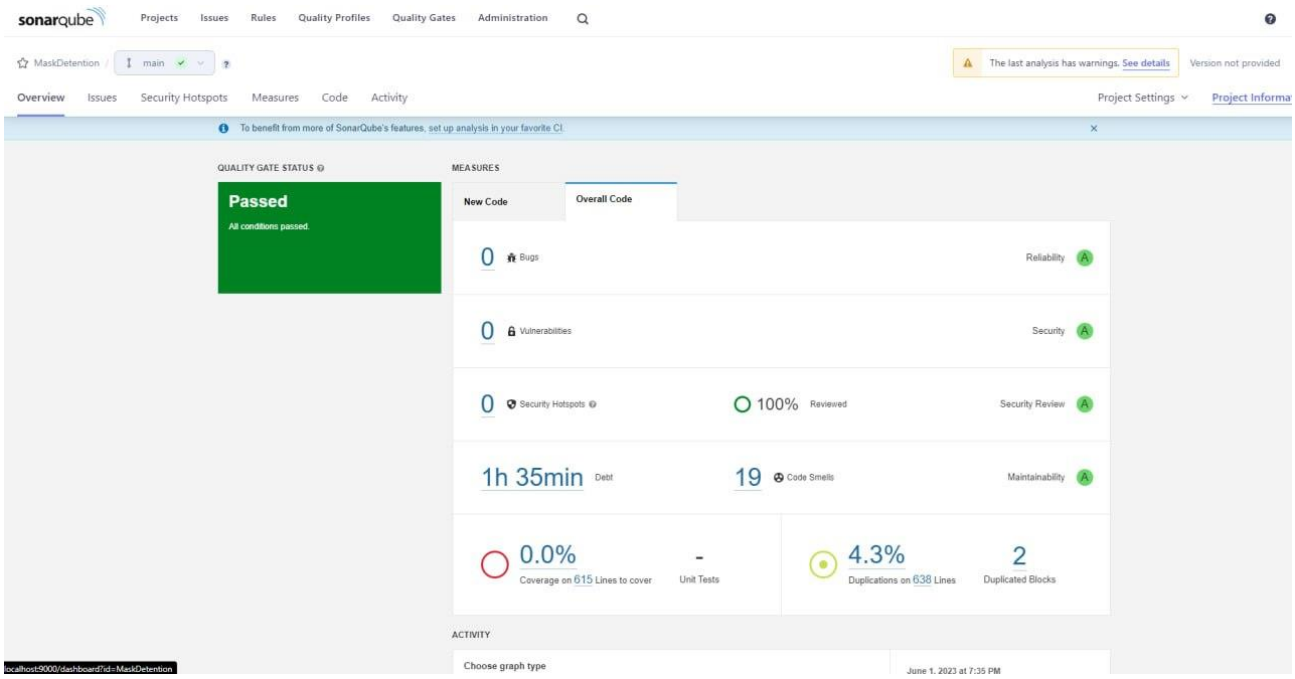

Рисунок 3.1 – Звіт SonarQube

Аналіз не виявив вразливостей, проте наявні певні структури або шаблони коду, які можуть призвести до складності розуміння коду, збільшення його складності, погіршенню його розширюваності. Проте на даний момент це ніяк не впливає на роботу коду.

3.2 Опис процесів тестування

Мануальні тести є важливою складовою частиною процесу тестування програмного забезпечення. Вони виконуються вручну, без використання автоматизованих скриптів або інструментів. Існує кілька причин, для яких проводяться мануальні тести.

По-перше, мануальні тести дають можливість оцінити користувальницький досвід. Людина, яка виконує тести, може зіграти роль кінцевого користувача і перевірити, наскільки програмне забезпечення задовольняє його потреби, чи працює інтерфейс коректно, чи немає помилок або неочікуваної поведінки. Мануальні тести дозволяють виявити проблеми, які можуть бути пропущені автоматизованими тестами.

По-друге, мануальні тести дозволяють провести складні або нестандартні сценарії тестування. Вони можуть включати в себе взаємодію з програмним забезпеченням у реальному часі, проведення складних операцій або введення некоректних даних. Людина, яка виконує мануальні тести, може активно досліджувати програму та шукати потенційні проблеми, що допомагає виявити баги, які автоматизовані тести можуть пропустити.

По-третє, мануальні тести можуть бути проведені в реальних умовах використання. Це може включати тестування програмного забезпечення на різних пристроях, платформах або мережевих умовах, що дає змогу перевірити його сумісність та поведінку в реальному середовищі. Мануальні тести також можуть допомогти виявити проблеми з продуктивністю, навантаженням або витривалістю програмного забезпечення.

Усі ці фактори роблять мануальні тести важливим етапом процесу тестування, який доповнює автоматизовані тести. Вони дозволяють забезпечити високу якість програмного забезпечення, виявити проблеми, що можуть бути пропущені автоматизованими засобами, та забезпечити задоволення користувачів від роботи з програмою.

В таблицях 3.1-3.8 описано основні тест-кейси, які були розглянуті під час мануального тестування.

Таблиця 3.1 – Тест-кейс 1.1

Тест	Авторизація користувача
Модуль	Авторизація користувача
Номер тесту	1.1
Початковий стан системи	Користувач на сторінці авторизації
Вхідні дані	Логін, пароль
Опис проведення тесту	Користувач вводить коректний логін та коректний пароль та натискає кнопку авторизації
Очікуваний результат	Авторизація проходить успішно та людина перенаправляється на головний екран
Фактичний результат	Авторизація проходить успішно та людина перенаправляється на головний екран

Таблиця 3.2 – Тест-кейс 1.2

Тест	Авторизація користувача
Модуль	Авторизація користувача
Номер тесту	1.2
Початковий стан системи	Користувач на сторінці авторизації
Вхідні дані	Логін, пароль
Опис проведення тесту	Користувач вводить некоректний логін або некоректний пароль та натискає кнопку авторизації
Очікуваний результат	Користувач отримує повідомлення про некоректність введених даних
Фактичний результат	Користувач отримує повідомлення про некоректність введених даних

Таблиця 3.3 – Тест-кейс 2.1

Тест	Реєстрація користувача
Модуль	Реєстрація користувача
Номер тесту	2.1
Початковий стан системи	Користувач на сторінці реєстрації
Вхідні дані	Логін, пароль

Продовження таблиці 3.3

Опис проведення тесту	Користувач вводить логін, який не використовується іншим користувачем, та пароль, який містить як мінімум 8 символів, як мінімум 1 цифру, як мінімум одну літеру верхнього регістру.
Очікуваний результат	Користувач отримує повідомлення про успішну реєстрацію
Фактичний результат	Користувач отримує повідомлення про успішну реєстрацію

Таблиця 3.4 – Тест-кейс 2.2

Тест	Реєстрація користувача
Модуль	Реєстрація користувача
Номер тесту	2.2
Початковий стан системи	Користувач на сторінці реєстрації
Вхідні дані	Логін, пароль
Опис проведення тесту	Користувач вводить логін, який вже використовується іншим користувачем та пароль, який містить як мінімум 8 символів, як мінімум 1 цифру, як мінімум одну літеру верхнього регістру.
Очікуваний результат	Користувач отримує повідомлення про те, що потрібно обрати інший логін
Фактичний результат	Користувач отримує повідомлення про те, що потрібно обрати інший логін

Таблиця 3.5 – Тест-кейс 3.1

Тест	Перегляд статистики
Модуль	Перегляд статистики
Номер тесту	3.1
Початковий стан системи	Користувач на сторінці перегляду статистики
Вхідні дані	Початкова дата, кінцева дата, наявність маски
Опис проведення тесту	Користувач обирає початкову дату, обирає кінцеву дату та обирає показати статистику з наявною маскою на обличчі чи без
Очікуваний результат	Користувач отримує актуальні дані статистики
Фактичний результат	Користувач отримує актуальні дані статистики

Таблиця 3.6 – Тест-кейс 4.1

Тест	Перегляд відеопотоку
Модуль	Перегляд відеопотоку
Номер тесту	4.1
Початковий стан системи	Користувач на екрані визначення наявності аксесуару на обличчі
Вхідні дані	Відеопоток з камери, на якому видно обличчя людини в масці
Опис проведення тесту	Людина знаходиться перед камерою у медичній масці
Очікуваний результат	Користувач бачить на екрані, що було визначено наявність маски у нього на обличчі
Фактичний результат	Користувач бачить на екрані, що було визначено наявність маски у нього на обличчі

Таблиця 3.7 – Тест-кейс 4.2

Тест	Перегляд відеопотоку
Модуль	Перегляд відеопотоку
Номер тесту	4.2
Початковий стан системи	Користувач на екрані визначення наявності аксесуару на обличчі
Вхідні дані	Відеопоток з камери, на якому видно обличчя людини без маски
Опис проведення тесту	Людина знаходиться перед камерою без медичної маски
Очікуваний результат	Користувач бачить на екрані, що було визначено відсутність маски у нього на обличчі
Фактичний результат	Користувач бачить на екрані, що було визначено відсутність маски у нього на обличчі

Таблиця 3.8 – Тест-кейс 4.3

Тест	Перегляд відеопотоку
Модуль	Перегляд відеопотоку
Номер тесту	4.3
Початковий стан системи	Користувач на екрані визначення наявності аксесуару на обличчі
Вхідні дані	Відеопоток з камери, на якому відсутня людина

Продовження таблиці 3.8

Опис проведення тесту	Людина не знаходиться перед камерою
Очікуваний результат	Програмне забезпечення продовжує працювати без помилок
Фактичний результат	Програмне забезпечення продовжує працювати без помилок

Окрім цього, було протестовано нейромережу. Тестування нейромережі YOLOv4 на тестових даних включало кілька кроків для оцінки її продуктивності та точності.

– Підготовка тестових даних: Спочатку було підготовано тестові дані для використання в процесі тестування. Важливо було забезпечити різноманітність тестових даних, включаючи різні типи об'єктів, розміри, кольори, фони та умови освітлення.

– Завантаження моделі: Наступним кроком було завантаження попередньо навченої моделі YOLOv4, яка була розроблена та навчена на тренувальних даних. Ця модель включає в себе ваги та архітектуру нейромережі, які використовуються для виявлення об'єктів на зображеннях.

– Виконання детекції об'єктів: Після завантаження моделі було виконано процес детекції об'єктів на тестових даних. Це означає передачу зображення або відео через нейромережу YOLOv4, яка аналізує зображення та виявляє об'єкти на ньому. Результатом було отримання координат та класів знайдених об'єктів.

– Оцінка результатів: Після детекції об'єктів на тестових даних було оцінено результати. Це включало порівняння з анотаціями тестових даних, щоб визначити, чи правильно нейромережа виявила та класифікувала об'єкти.

У результаті тестування нейромережі YOLOv4 на тестових даних отримано інформацію про її ефективність та надійність у виявленні об'єктів

на реальних зображеннях або відео. Цей процес дозволяє перевірити, наскільки добре нейромережа виконує своє завдання та чи відповідає вона вимогам проєкту.

3.3 Опис контрольного прикладу

Для більш ретельного тестування вирішено провести тестування з врахуванням всіх можливих розгалужень. Цей процес відбувається наступним чином:

– Реєстрація (Рисунок 3.2)

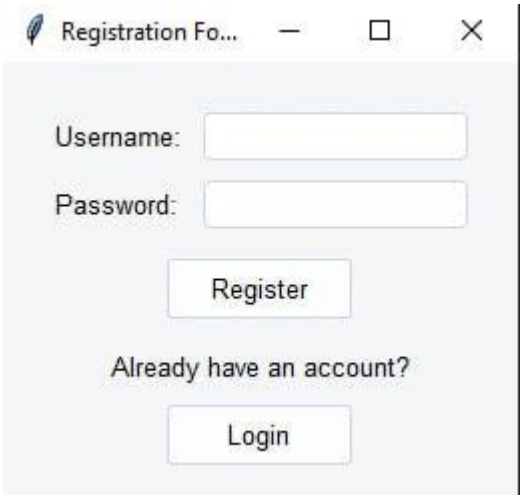

Рисунок 3.2 – Реєстрація

– Авторизація (Рисунок 3.3)

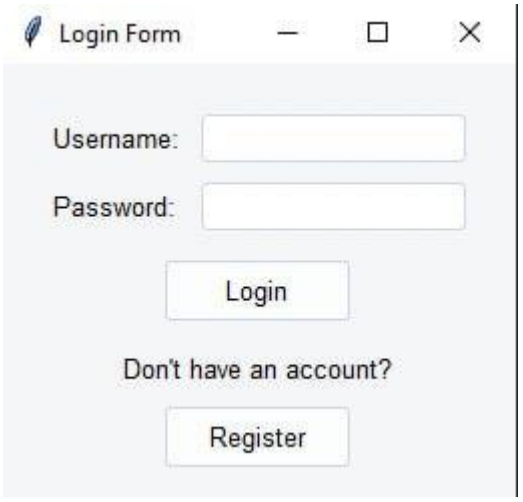

Рисунок 3.3 – Авторизація

– Головне меню (Рисунок 3.4)

Рисунок 3.4 – Головне меню

– Перегляд статистики (Рисунок 3.5 та Рисунок 3.6)

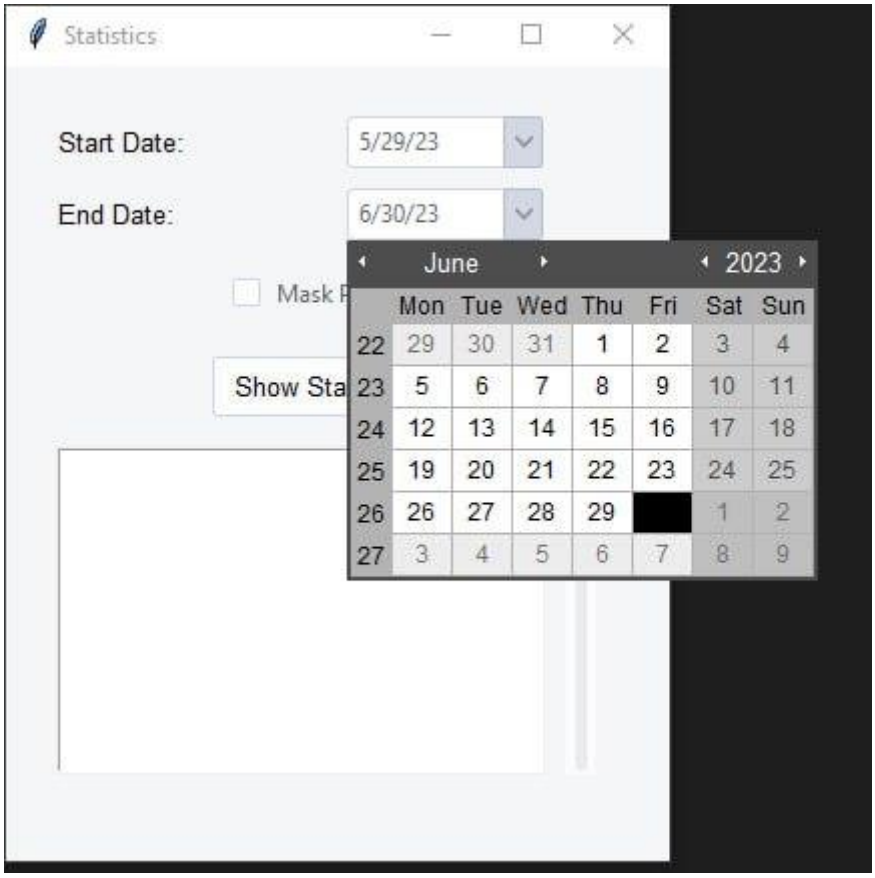

Рисунок 3.5 – Вибір дати у вікні статистики

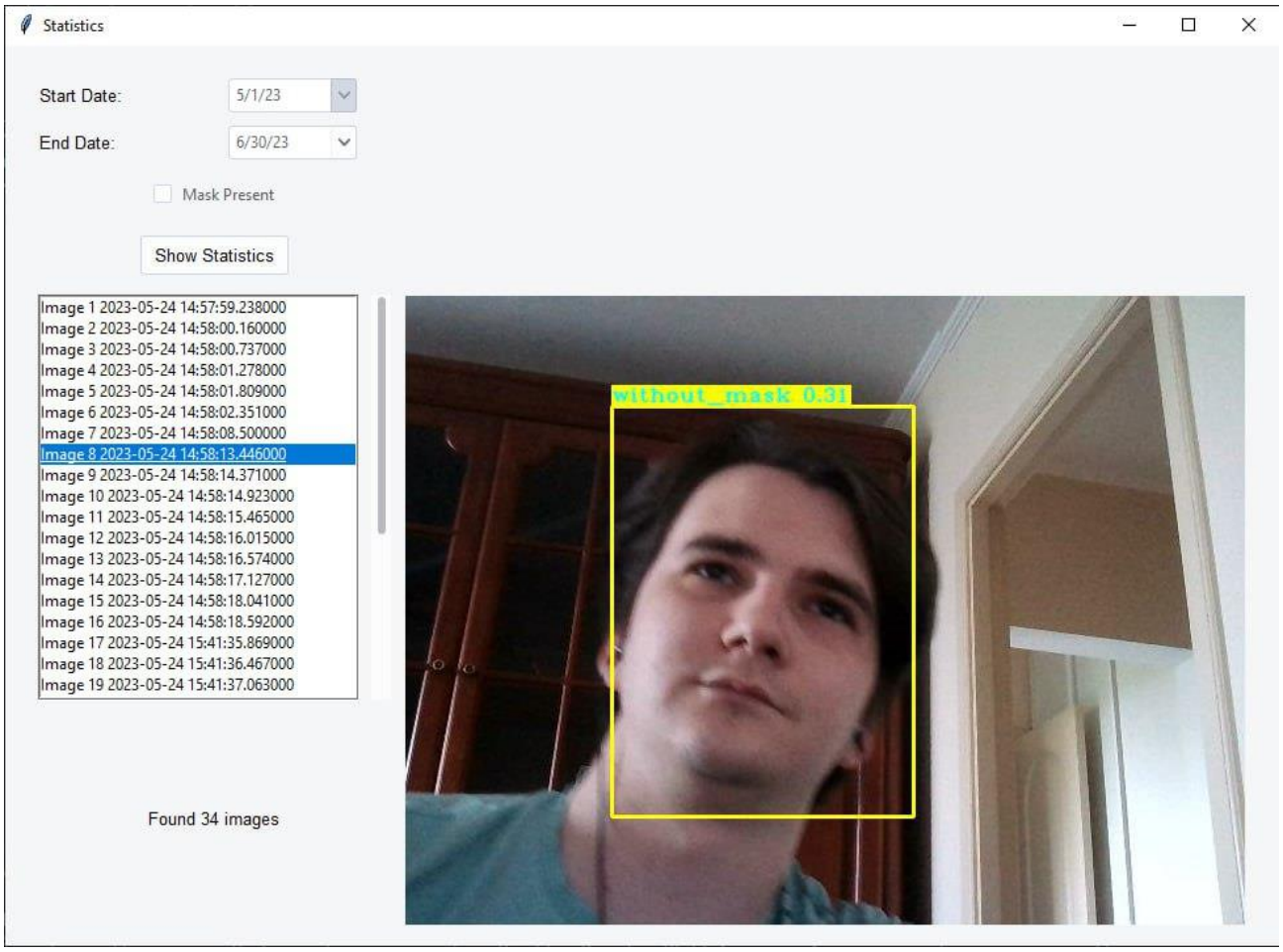

Рисунок 3.6 – Перегляд статистики та зображень

– Перегляд наявності маски (Рисунок 3.7)

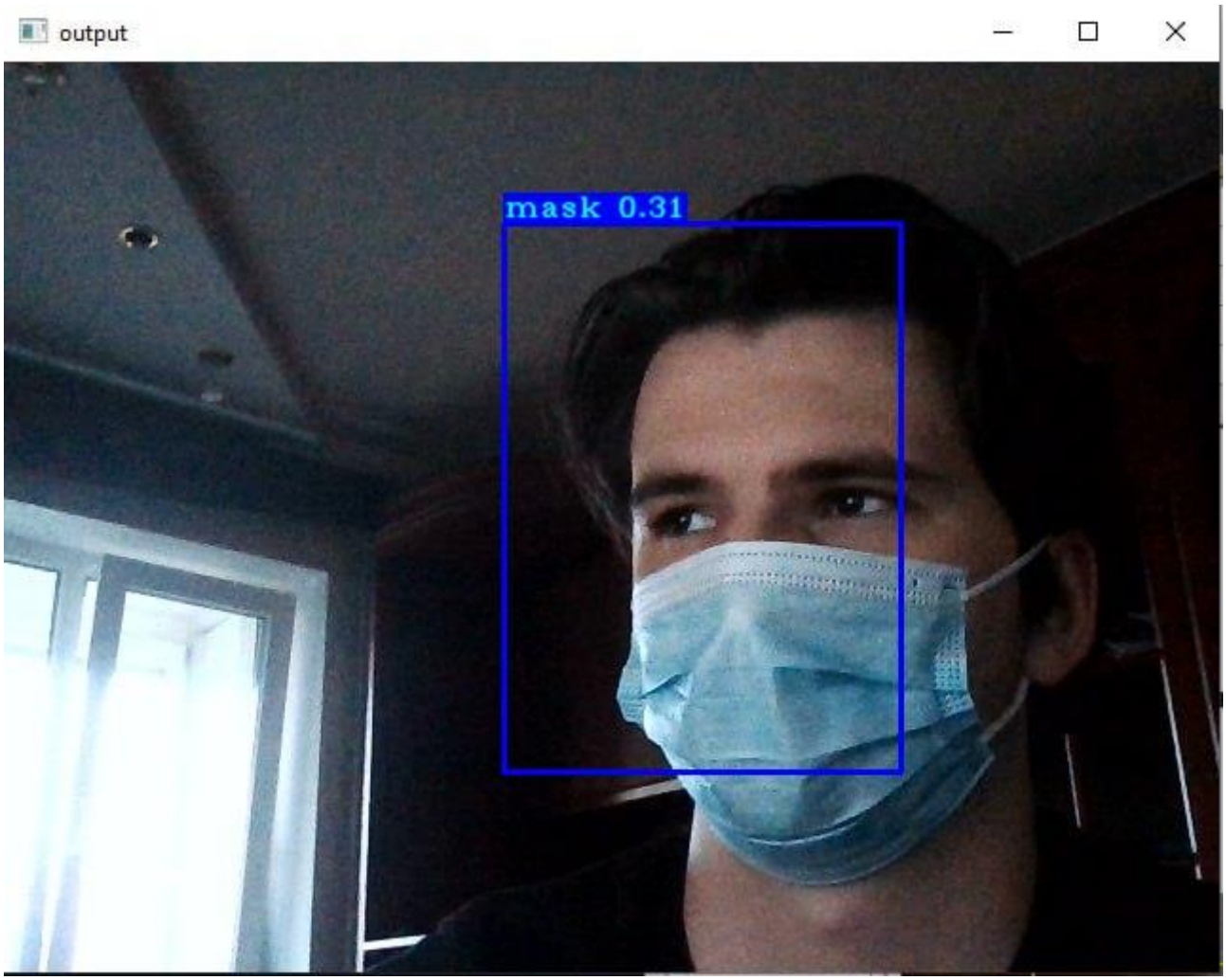

Рисунок 3.7 – Перегляд наявності маски

Висновки до розділу

Після проведення аналізу та реалізації програмного забезпечення, відбулось тестування, яке мало на меті перевірити якість та коректність роботи розробленого продукту.

У цьому розділі використано інструмент SonarQube для аналізу якості коду. SonarQube дозволив виявити потенційні проблеми та дефекти в коді, такі як проблеми з безпекою та стиль коду. Це дозволило отримати повний огляд стану кодової бази та рекомендації щодо поліпшення якості програмного забезпечення.

Другим етапом було мануальне тестування, де перевірено функціональність та коректність роботи програми. Виконувались різні сценарії, щоб переконатись, що програма правильно виявляє наявність

захисної маски на обличчі людини. Це включало тести з різними типами зображень або відео, різними умовами освітлення та перспективами, а також перевірку правильної класифікації обличь без масок та з масками.

Також зроблено опис контрольного прикладу, з усіма можливими розгалуженнями.

Розділ про тестування допоміг забезпечити високу якість, надійність та безпеку розробленого програмного забезпечення, а також виявити та виправити можливі проблеми. Цей етап підтверджує, що програма працює належним чином, задовольняє вимоги та очікування користувачів та готова до подальшого впровадження.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Початковим етапом розгортання програмного забезпечення є підготовка оточення для розгортання. Це включає установку необхідних залежностей, таких як Python та необхідні бібліотеки для нейромережі YOLO4 та взаємодії з базою даних MongoDB. Для десктопного застосунку можуть знадобитися додаткові компоненти, залежно від конкретних вимог та платформи.

Наступним етапом є підготовка моделі нейромережі YOLO4 та її інтеграція в програмне забезпечення. Це включає завантаження та налаштування вагових файлів моделі, визначення правил виявлення маски на обличчі, а також написання коду для взаємодії з моделлю та обробки результатів.

Третім етапом є підготовка бази даних MongoDB та її інтеграція з програмним забезпеченням. Це включає створення необхідних колекцій та схем даних, встановлення з'єднання з базою даних та написання коду для зберігання та отримання даних про виявлені обличчя з масками.

Останнім етапом є розгортання десктопного застосунку. Це може включати компіляцію програми, створення виконуваного файлу або пакету для встановлення, налаштування інтерфейсу користувача та забезпечення доступу до нейромережі та бази даних. Після успішного розгортання десктопний застосунок готовий до використання для перевірки наявності масок на обличчях.

Цей процес розгортання дозволяє ефективно використовувати нейромережу YOLO4 для перевірки наявності масок на обличчях, використовуючи базу даних MongoDB для зберігання результатів. Десктопний застосунок на пітоні надає зручний інтерфейс для користувачів, що дозволяє легко та швидко проводити перевірку.

Структура проекту наведена на Рисунку 4.1.

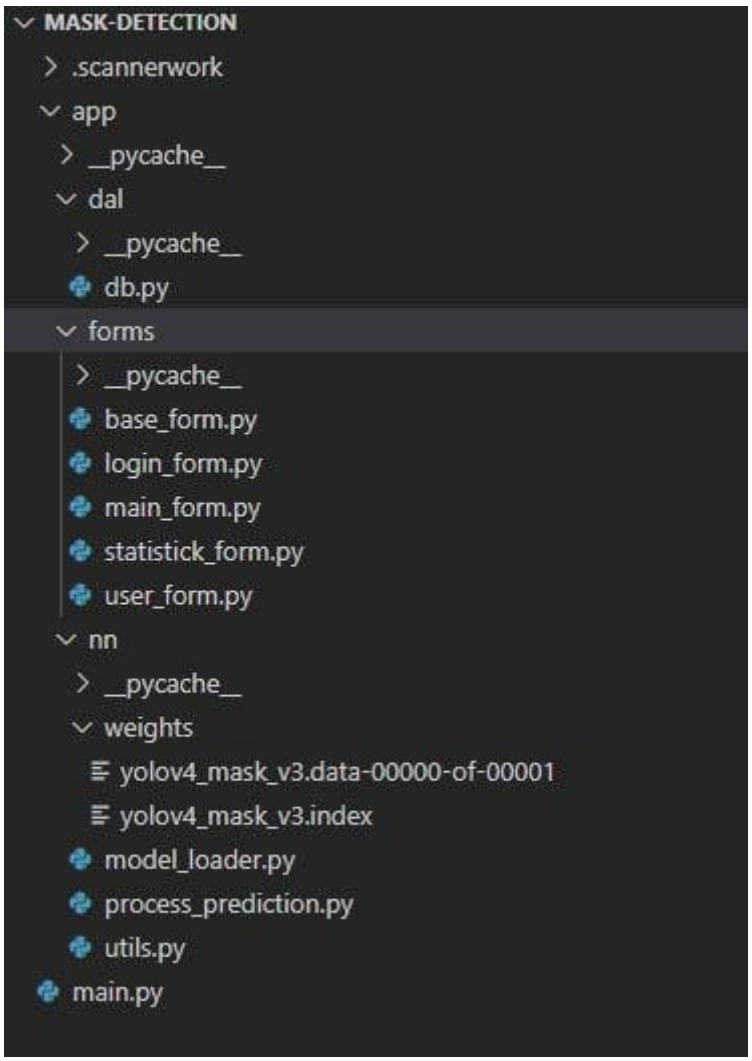

Рисунок 4.1 – Структура проєкту

4.2 Підтримка програмного забезпечення

Першим етапом підтримки є моніторинг та відлагодження системи. Це включає постійне спостереження за роботою програмного забезпечення, перевірку його стабільності та виявлення можливих проблем. Крім того, важливо періодично оновлювати нейромережу YOLO4 та необхідні бібліотеки, а також забезпечувати сумісність з актуальною версією бази даних MongoDB.

Другим етапом є вирішення проблем та внесення змін. Підтримка включає в себе виявлення та аналіз виниклих проблем, таких як помилки в роботі нейромережі, проблеми з базою даних або інтерфейсом користувача.

На основі цього проводяться необхідні корекції та виправлення, щоб забезпечити правильну та безперебійну роботу програмного забезпечення.

Третім етапом є надання технічної підтримки користувачам. Це включає відповіді на запитання та допомогу щодо використання десктопного застосунку, нейромережі та бази даних. Технічна підтримка може здійснюватися через різні канали, такі як електронна пошта, телефонні дзвінки або онлайн-чати, з метою надання швидких та ефективних рішень для користувачів.

Всі ці етапи підтримки спрямовані на забезпечення безперебійної та ефективної роботи програмного забезпечення, що містить нейромережу для перевірки наявності маски на обличчі людини, з використанням бази даних MongoDB та десктопного застосунку на пітоні.

Висновки до розділу

В даному розділі описано процес розгортання програмного забезпечення на основі нейронної мережі. На початковому етапі була виконана підготовка оточення для розгортання програмного забезпечення, включаючи установку необхідних залежностей. Наступним етапом була підготовка та інтеграція моделі нейромережі YOLO4 в програмне забезпечення, зокрема завантаження вагових файлів та налаштування правил виявлення масок на обличчі. Третім етапом була підготовка бази даних MongoDB та її інтеграція з програмним забезпеченням, включаючи створення колекцій та схем даних. На останньому етапі було розгорнуто десктопний застосунок, включаючи компіляцію, налаштування інтерфейсу та забезпечення доступу до нейромережі та бази даних. В результаті цих кроків була створена ефективна система для перевірки наявності масок на обличчях, яка працює зручно та надійно для користувачів.

Також описано процес підтримки програмного забезпечення, а саме постійний моніторинг, відлагодження та оновлення системи для

забезпечення стабільності й сумісності з актуальними компонентами, вирішення помилок та проблем.

ВИСНОВКИ

Початковим етапом роботи над дипломним проєктом було проведення глибокого аналізу предметної області, використовуючи різноманітні джерела і дані. Виявлено основні проблеми цієї області і знайдено потенційні шляхи їх вирішення. Також проаналізовано існуюче алгоритмічне забезпечення і технологічні рішення, що вже використовуються, для визначення тих з них, які можуть бути використані у програмному забезпеченні.. На основі цього аналізу були сформульовані функціональні та нефункціональні вимоги до програмного забезпечення.

Наступним кроком було виявлення ключових бізнес-процесів, які пов'язані з розроблюваним продуктом, і проведено їх детальний аналіз. Далі було з’ясовано, як компоненти програмного забезпечення взаємодіють між собою і створено структуру продукту для подальшого проєктування і розробки. Це моделювання сприяло вибору архітектури програмного забезпечення. Після цього розпочалася фаза конструювання продукту, в рамках якої було спроєктовано нейромережу для визначення наявності аксесуарів, зокрема масок, на обличчі людини.

Після завершення конструювання продукту, було проведено аналіз якості за допомогою SonarQube, мануальне тестування функціональності та обробка сценаріїв з різними розгалуженнями. Тестування допомогло забезпечити високу якість, надійність та безпеку продукту.

Після успішного описано процес розгортання програмного забезпечення включаючи підготовку оточення, інтеграцію нейромережі та бази даних, а також розгортання десктопного застосунку. На цьому етапі були проаналізовані інструменти для розгортання програмного забезпечення, заснованого на нейронній мережі, та детально описаний процес розгортання продукту з можливістю його підтримки.

В результаті виконання дипломного проєкту було спроєктовано програмне забезпечення, яке здатне визначати наявність аксесуарів, зокрема

масок, на обличчі людини, та надавати можливість переглядати статистику та оброблені нейромережею зображення. Програмне забезпечення було розроблене на основі побудованої нейронної мережі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Global research on coronavirus disease (COVID-19). *World Health Organization.* URL: https://www.who.int/emergencies/diseases/novel-coronavirus-2019/global-research-on-novel-coronavirus-2019-ncov (дата звернення: 11.04.2023).

2. Коронавірус: статистика по країнах Європи. *Мінфін.* URL: https://index.minfin.com.ua/ua/reference/coronavirus/geography/europe/ (дата звернення 18.05.2023).

3. Коронавірус: Шляхи передачі та способи захисту. *Центральна поліклініка Міністерства внутрішніх справ України.* URL: https://cp.mvs.gov.ua/attention (дата звернення 18.04.2023).

4. Advice on the use of masks in the context of COVID-19: interim guidance, 2020. *World Health Organization.* URL: https://apps.who.int/iris/handle/10665/332293 (дата звернення 18.04.2023).

5. Understanding SSD MultiBox – Real-Time Object Detection In Deep Learning. *Medium.* URL: https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab (дата звернення 20.04.2023).

6. Image Classification With MobileNet. *BuiltIn.* URL: https://builtin.com/machine-learning/mobilenet (дата звернення 20.04.2023).

7. What is YOLOv4? A Detailed Breakdown. *Roboflow.* URL: https://blog.roboflow.com/a-thorough-breakdown-of-yolov4/. (дата звернення 20.04.2023).

8. Official Python documentation. *Python.* URL: https://www.python.org/ (дата звернення 29.04.2023).

9. Official R documentation. *R-Project.* URL: https://www.r-project.org/other-docs.html (дата звернення 29.04.2023).

10. Official Java documentation. *Oracle.* URL: https://docs.oracle.com/en/java/ (дата звернення 29.04.2023).

11. MaskCheck. *GetMaskCheck*. URL: https://getmaskcheck.com/ (дата звернення 07.05.2023).

12. SafeSpace App. *Google Play*. URL: https://play.google.com/store/apps/details?id=com.saferwork.safetyfirst&hl=uk&gl=US&pli=1 (дата звернення 07.05.2023).

13. Functional and Nonfunctional Requirements: Specification and Types. *Altexsoft*. URL: https://www.altexsoft.com/blog/business/functional-and-non-functional-requirements-specification-and-types/. (дата звернення 15.07.2023).

ДОДАТОК А ЗВІТ ПОДІБНОСТІ

UNICHECK

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1015409929

Дата перевірки:
04.06.2023 07:41:43 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
04.06.2023 07:46:32 EEST

ID користувача:
76913

Назва документа: **IT-93_Григоренко_ПЗ**

Кількість сторінок: **60** Кількість слів: **8843** Кількість символів: **70329** Розмір файлу: **4.44 MB** ID файлу: **1015073288**

10.6%
Схожість

Найбільша схожість: **5.34%** з джерелом з Бібліотеки (ID файлу: **1015069347**)

3.58% Джерела з Інтернету	145	Сторінка 62
10.2% Джерела з Бібліотеки	255	Сторінка 64

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ НАЯВНОСТІ АКСЕСУАРІВ НА ОБЛИЧЧІ ЛЮДИНИ

Текст програми

КПІ.ІП-9308.045490.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Ярослав ГРИГОРЕНКО

Київ – 2023

Файл db.py

```
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi
from PIL import Image
import io
import numpy as np
import gridfs
from datetime import datetime
import cv2
uri = "mongodb+srv://user1:CqkUDCFvNfMN4aOM@mask-
detectin.uzpyxfr.mongodb.net/?retryWrites=true&w=majority"
# Create a new client and connect to the server
client = MongoClient(uri, server_api=ServerApi("1"))
db = client["mask-detectin"]
fs = gridfs.GridFS(db)
users_collection = db["users"]
def test_connection():
    try:
        client.admin.command("ping")
        print("Pinged your deployment. You successfully connected to MongoDB!")
    except Exception as e:
        print(e)
def create_user(username, password):
    # Check if the username already exists
    if users_collection.find_one({"username": username}):
        raise ValueError("Username already exists")
    # Create a new user document in the users collection
    user = {"username": username, "password": password}
    users_collection.insert_one(user)
def find_user(username, password):
```

```
    # Check if the username and password match
    user = users_collection.find_one({"username": username, "password":
password})
    return user
def get_all_users():
    users = users_collection.find({})
    return users
def delete_user(username):
    user_images = fs.find({"filename": {"$regex": f"^{username}_.*\.jpg$"}})
    for image in user_images:
        fs.delete(image._id)
    users_collection.delete_one({"username": username})
def save_image_to_db(image, username, filename, mask_present):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = Image.fromarray(np.uint8(image)).convert("RGB")
    byte_arr = io.BytesIO()
    image.save(byte_arr, format="JPEG")
    byte_arr = byte_arr.getvalue()
    fs.put(byte_arr, filename=f"{username}_{filename}.jpg",
uploadDate=datetime.utcnow(), mask=mask_present)
def get_images_from_db(username, start_time, end_time, mask_present):
    files = fs.find(
        {
            "filename": {"$regex": f"^{username}_"},
            "uploadDate": {"$gte": start_time, "$lt": end_time},
            "mask": mask_present,
        }
    )
    images = []
    for file in files:
```

```
        image = Image.open(io.BytesIO(file.read()))
        upload_date = file.uploadDate  # Extract the upload date of the image
        images.append((image, upload_date))  # Append a tuple of the image and its
upload date
    return images
def get_all_images_from_db(username, start_time, end_time, mask_present):
    files = fs.find(
        {
            "uploadDate": {"$gte": start_time, "$lt": end_time},
            "mask": mask_present,
        }
    )
    images = []
    for file in files:
        image = Image.open(io.BytesIO(file.read()))
        upload_date = file.uploadDate  # Extract the upload date of the image
        images.append((image, upload_date))  # Append a tuple of the image and its
upload date
    return images
```

Файл base_form.py

```
import tkinter as tk
from tkinter import messagebox, ttk
class Form(ttk.Frame):
    def __init__(self, parent):
        super().__init__(parent)
        self.parent = parent
    def show_message(self, text):
        messagebox.showinfo("Message", text)
    def clear_fields(self):
        pass
```

Файл login_form.py

```
import tkinter as tk
from tkinter import ttk
from ttkthemes import ThemedTk
from app.dal.db import create_user
from app.forms.base_form import Form
from app.forms.main_form import MainForm
from app.dal.db import find_user
class LoginForm(Form):
    def __init__(self, parent):
        self.parent = parent
        self.parent.title("Login Form")
        self.style = ttk.Style()
        self.style.configure('TButton', font=('Arial', 10), foreground='black')
        self.style.configure('TLabel', font=('Arial', 10), foreground='black')
        self.style.configure('TEntry', font=('Arial', 10), foreground='black')
        self.content_frame = ttk.Frame(self.parent, padding=(20, 20, 20, 10))
        self.content_frame.pack()
        self.username_label = ttk.Label(self.content_frame, text="Username:")
        self.username_label.grid(row=0, column=0, sticky="w", padx=5, pady=5)
        self.username_entry = ttk.Entry(self.content_frame)
        self.username_entry.grid(row=0, column=1, padx=5, pady=5)
        self.password_label = ttk.Label(self.content_frame, text="Password:")
        self.password_label.grid(row=1, column=0, sticky="w", padx=5, pady=5)
        self.password_entry = ttk.Entry(self.content_frame, show="*")
        self.password_entry.grid(row=1, column=1, padx=5, pady=5)
        self.login_button = ttk.Button(self.content_frame, text="Login",
command=self.login)
        self.login_button.grid(row=2, column=0, columnspan=2, pady=10)
```

```
        self.register_text = ttk.Label(self.content_frame, text="Don't have an
account?")
        self.register_text.grid(row=3, column=0, columnspan=2, pady=5)
        self.register_button = ttk.Button(self.content_frame, text="Register",
command=self.show_registration_form)
        self.register_button.grid(row=4, column=0, columnspan=2, pady=5)
    def login(self):
        username = self.username_entry.get()
        password = self.password_entry.get()
        # Check user credentials in the database
        user = find_user(username, password)
        if user:
            self.clear_fields()
            self.parent.destroy()
            main_root = ThemedTk(theme="arc")
            main_form = MainForm(main_root, username, "admin" if
username=="admin" else "client")
            main_root.mainloop()
        else:
            self.show_message("Invalid username or password")
    def show_registration_form(self):
        self.clear_fields()
        self.parent.destroy()
        registration_root = ThemedTk(theme="arc")
        registration_form = RegistrationForm(registration_root)
        registration_root.mainloop()
class RegistrationForm(Form):
    def __init__(self, parent):
        self.parent = parent
        self.parent.title("Registration Form")
```

```
        self.style = ttk.Style()
        self.style.configure('TButton', font=('Arial', 10), foreground='black')
        self.style.configure('TLabel', font=('Arial', 10), foreground='black')
        self.style.configure('TEntry', font=('Arial', 10), foreground='black')
        self.content_frame = ttk.Frame(self.parent, padding=(20, 20, 20, 10))
        self.content_frame.pack()
        self.username_label = ttk.Label(self.content_frame, text="Username:")
        self.username_label.grid(row=0, column=0, sticky="w", padx=5, pady=5)
        self.username_entry = ttk.Entry(self.content_frame)
        self.username_entry.grid(row=0, column=1, padx=5, pady=5)
        self.password_label = ttk.Label(self.content_frame, text="Password:")
        self.password_label.grid(row=1, column=0, sticky="w", padx=5, pady=5)
        self.password_entry = ttk.Entry(self.content_frame, show="*")
        self.password_entry.grid(row=1, column=1, padx=5, pady=5)
        self.register_button = ttk.Button(self.content_frame, text="Register",
command=self.register)
        self.register_button.grid(row=2, column=0, columnspan=2, pady=10)
        self.login_text = ttk.Label(self.content_frame, text="Already have an
account?")
        self.login_text.grid(row=3, column=0, columnspan=2, pady=5)
        self.login_button = ttk.Button(self.content_frame, text="Login",
command=self.show_login_form)
        self.login_button.grid(row=4, column=0, columnspan=2, pady=5)
    def register(self):
        username = self.username_entry.get()
        password = self.password_entry.get()
        try:
            # Create a new user in the database
            create_user(username, password)
            self.show_message("Registration successful!")
```

```
            self.clear_fields()
        except ValueError:
            self.show_message("Username already exists. Please choose a different
username.")
    def show_login_form(self):
        self.clear_fields()
        self.parent.destroy()
        login_root = ThemedTk(theme="arc")
        login_form = LoginForm(login_root)
        login_root.mainloop()
```

Файл main_form.py

```
import datetime
import tkinter as tk
from tkinter import ttk
from PIL import Image, ImageTk
from app.forms.base_form import Form
from app.forms.statistick_form import StatisticsForm
from app.forms.user_form import UserForm
from app.nn.model_loader import load_model
from app.nn.process_prediction import start_prediction_video
model =  load_model("yolov4_mask_v3")
class MainForm:
    def __init__(self, parent, username, role):
        self.parent = parent
        self.username = username
        self.role = role
        self.parent.geometry("200x250")
        self.parent.title("Main Form")
        self.parent.configure(background='white')
        self.style = ttk.Style()
```

```
        self.style.configure('TButton', font=('Arial', 10), foreground='black')
        self.style.configure('TLabel', font=('Arial', 10), foreground='black')
        self.content_frame = ttk.Frame(self.parent, padding=(20, 20, 20, 10))
        self.content_frame.pack()
        self.label = ttk.Label(self.content_frame, text=f"Welcome, {self.username}!")
        self.label.grid(row=0, column=0, columnspan=2, pady=10)
        self.start_button = ttk.Button(self.content_frame, text="Start Prediction",
command=lambda: self.start_prediction())
        self.start_button.grid(row=1, column=0, columnspan=2, pady=10)
        self.stats_button = ttk.Button(self.content_frame, text="Open Statistics",
command=lambda: self.open_statistics())
        self.stats_button.grid(row=2, column=0, columnspan=2, pady=10)
        if self.role == "admin":
            self.users_button = ttk.Button(self.content_frame, text="Users
Managment", command=lambda: self.open_user_managment())
            self.users_button.grid(row=3, column=0, columnspan=2, pady=10)
    def start_prediction(self):
        start_prediction_video(model, self.username)
    def open_statistics(self):
        self.stats_window = StatisticsForm(self.parent, self.username, self.role)
    def open_user_managment(self):
        self.users_window = UserForm(self.parent)
```

Файл statistick_form.py

```
import tkinter as tk
from tkinter import ttk
from PIL import ImageTk, Image
import datetime
from tkcalendar import DateEntry
from app.dal.db import get_images_from_db, get_all_images_from_db
class StatisticsForm:
```

```
    def __init__(self, parent, username,role):
        self.parent = parent
        self.username = username
        self.role = role
        self.style = ttk.Style()
        self.style.configure('TButton', font=('Arial', 10), foreground='black')
        self.style.configure('TLabel', font=('Arial', 10), foreground='black')
        self.style.configure('TEntry', font=('Arial', 10), foreground='black')
        self.window = tk.Toplevel(self.parent)
        self.window.title("Statistics")
        self.content_frame = ttk.Frame(self.window, padding=(20, 20, 20, 10))
        self.content_frame.pack()
        self.start_date_label = ttk.Label(self.content_frame, text="Start Date:")
        self.start_date_label.grid(row=0, column=0, sticky="w", padx=5, pady=5)
        self.start_date = DateEntry(self.content_frame)
        self.start_date.grid(row=0, column=0, padx=(150,5), pady=5, sticky="w")
        self.end_date_label = ttk.Label(self.content_frame, text="End Date:")
        self.end_date_label.grid(row=1, column=0, sticky="w", padx=5, pady=5)
        self.end_date = DateEntry(self.content_frame)
        self.end_date.grid(row=1, column=0, padx=(150,5), pady=5, sticky="w")
        self.mask_present = tk.BooleanVar()  # Create a BooleanVar to track the state
of the Checkbutton
        self.mask_check = ttk.Checkbutton(self.content_frame, text="Mask Present",
variable=self.mask_present)
        self.mask_check.grid(row=2, column=0, columnspan=2, pady=10)
        self.show_button = ttk.Button(self.content_frame, text="Show Statistics",
command=lambda: self.show_statistics())
        self.show_button.grid(row=3, column=0, columnspan=2, pady=10)
        self.image_list = tk.Listbox(self.content_frame, width=30)
        self.image_list.grid(row=4, column=0, sticky='nsew', padx=5, pady=5)
```

```
        self.scrollbar = ttk.Scrollbar(self.content_frame, orient='vertical',
command=self.image_list.yview)
        self.scrollbar.grid(row=4, column=1, sticky='nsw', padx=5, pady=5)
        self.image_list.configure(yscrollcommand=self.scrollbar.set)
        self.image_label = ttk.Label(self.content_frame)
        self.image_label.grid(row=4, column=2, rowspan=2, padx=5, pady=5)
        self.image_count_label = ttk.Label(self.content_frame)
        self.image_count_label.grid(row=5, column=0, columnspan=2, padx=5,
pady=5)
    def show_statistics(self):
        start_time = datetime.datetime.combine(self.start_date.get_date(),
datetime.time())
        end_time = datetime.datetime.combine(self.end_date.get_date(),
datetime.time())
        mask_present = self.mask_present.get()
        get_images_func = get_all_images_from_db if self.role == "admin" else
get_images_from_db
        images = get_images_func(self.username, start_time, end_time,
mask_present)
        self.image_list.delete(0, tk.END)
        for i, (image, date) in enumerate(images):
            self.image_list.insert(tk.END, f"Image {i+1} {date}")
            self.image_list.bind('<<ListboxSelect>>', self.on_select)
        self.images = images
        self.image_count_label.config(text=f"Found {len(images)} images")
    def on_select(self, evt):
        index = self.image_list.curselection()[0]
        image = self.images[index][0]
        image = ImageTk.PhotoImage(image)
        self.image_label.config(image=image)
```

```
        self.image_label.image = image
```

Файл user_form.py

```
import tkinter as tk
from tkinter import messagebox
from app.dal.db import get_all_users, delete_user
from tkinter import ttk
class UserForm:
    def __init__(self, parent):
        self.parent = parent
        self.style = ttk.Style()
        self.style.configure('TButton', font=('Arial', 10), foreground='black')
        self.style.configure('TLabel', font=('Arial', 10), foreground='black')
        self.style.configure('TEntry', font=('Arial', 10), foreground='black')
        self.window = tk.Toplevel(self.parent)
        self.window.title("Statistics")
        self.content_frame = ttk.Frame(self.window, padding=(20, 20, 20, 10))
        self.content_frame.pack()
        self.window.geometry("200x300")
        self.window.title("User Form")
        self.parent.configure(background='white')
        self.window.configure(background='white')
        # Listbox
        self.listbox = tk.Listbox(self.content_frame)
        self.listbox.pack(pady=20)
        # Delete button
        self.delete_button = tk.Button(self.content_frame, text="Delete User",
command=self.remove_user)
        self.delete_button.pack(pady=20)
        self.load_users()
    def load_users(self):
```

```
    users = get_all_users()
    for user in users:
        if user["username"] != "admin":
            self.listbox.insert(tk.END, user["username"])
def remove_user(self):
    selected_user = self.listbox.get(tk.ANCHOR)
    if selected_user:
        delete_user(selected_user)
        self.listbox.delete(tk.ANCHOR)
        messagebox.showinfo("Success", "User deleted successfully")
    else:
        messagebox.showerror("Error", "No user selected")
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ НАЯВНОСТІ АКСЕСУАРІВ НА ОБЛИЧЧІ ЛЮДИНИ

Програма та методика тестування

КПІ.ІП-9308.045490.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Ярослав ГРИГОРЕНКО

Київ – 2023

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є програмне забезпечення для розпізнавання наявності маски на обличчі людини.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

– перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;

– перевірка збереження даних;

– перевірка сумісності застосунку з операційною системою Windows;

– 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;

– динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;

– 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.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з використанням наскрізного тестування та написанням unit-тестів, 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:

– динамічне тестування на відповідність функціональним вимогам;

– тестування на виведення повідомлень про помилку, коли це необхідно;

– тестування інтерфейсу користувача;

– тестування зручності використання.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ НАЯВНОСТІ АКСЕСУАРІВ НА ОБЛИЧЧІ ЛЮДИНИ

Керівництво користувача

КПІ.ІП-9308.045490.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____ Максим ГОЛОВЧЕНКО

Виконавець:
_____ Ярослав ГРИГОРЕНКО

Київ – 2023

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Розробка призначена для розпізнавання наявності медичної маски на обличчі людини.

Метою розробки є покращення безпеки, за допомогою відстеження в медичних та державних установах визначення наявності аксесуарів на обличчі людини на основі нейронної мережі.

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

– наявність доступу до Інтернету;
– наявність середовища Python на персональному комп’ютері користувача.

2.2 Завантаження застосунку

На даний момент застосунок можна запустити власноруч, використовуючи відповідний exe-файл. Для цього спершу необхідно завантажити його на персональний комп’ютер, а потім виконати запуск даного додатку.

2.3 Перевірка коректної роботи

По завершенню запуску додатка повинна відобразитись початкова сторінка застосунку, а саме сторінка реєстрації.

3 ВИКОНАННЯ ПРОГРАМИ

При запуску програмного застосунку користувачу буде відображено вікно авторизації (Рисунок 3.1).

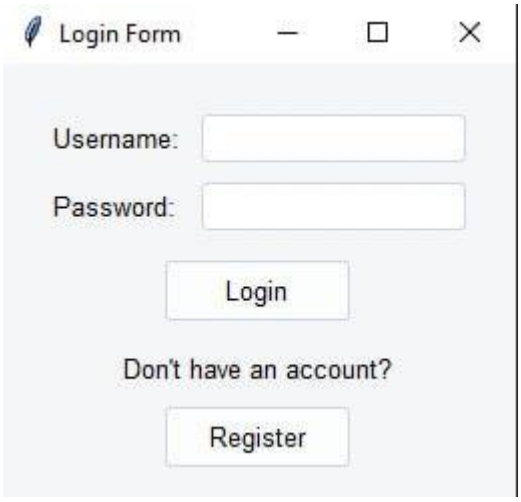

Рисунок 3.1 – Вікно авторизація

Якщо користувач не має акаунту в застосунку, він має можливість зареєструватися, перейшовши на вікно реєстрації (Рисунок 3.2).

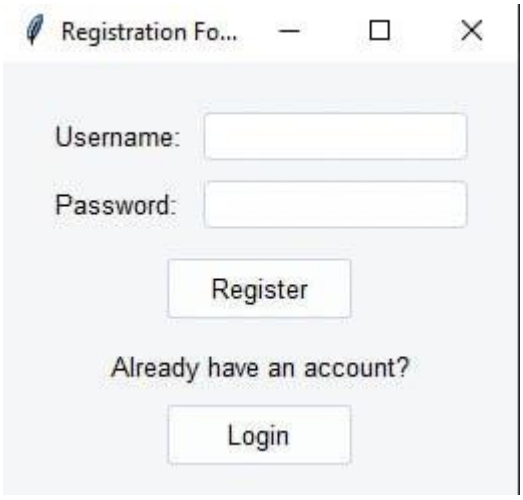

Рисунок 3.2 – Вікно реєстрації

Після успішної авторизації, користувач потрапляє на головне вікно (Рисунок 3.3).

Рисунок 3.3 – Головне вікно

З головного вікна можна пере до перегляду статистики (Рисунок 3.4) або до розпізнавання наявності аксесуарів на обличчі людини на відео (Рисунок 3.5).

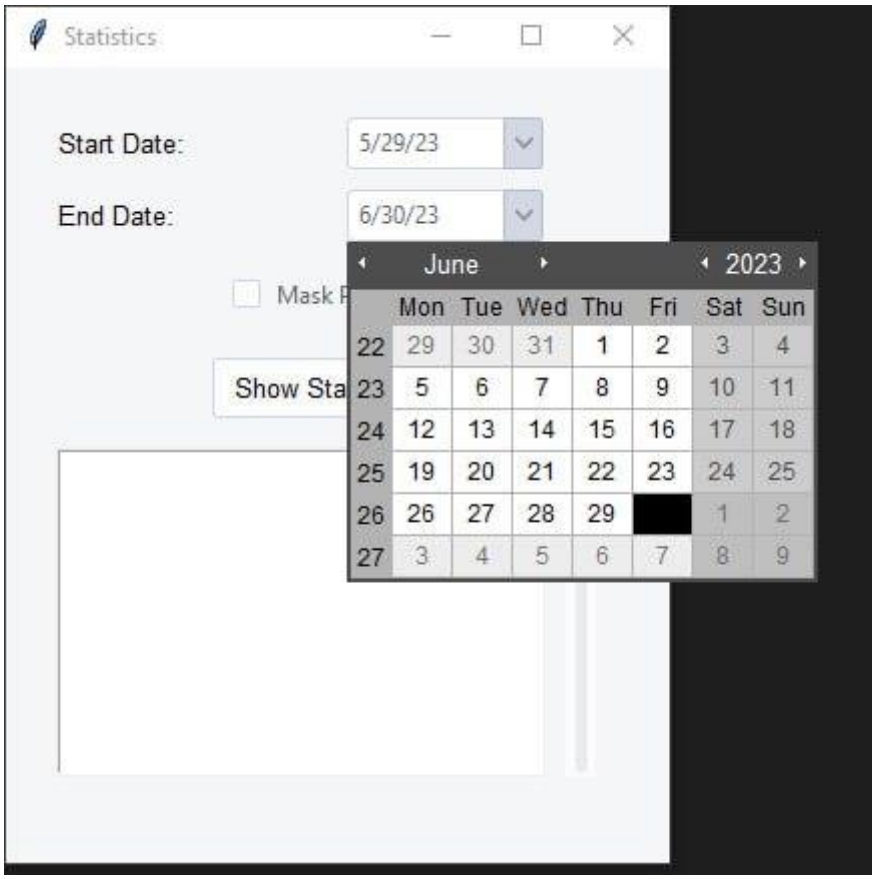

Рисунок 3.4 – Перегляд статистики

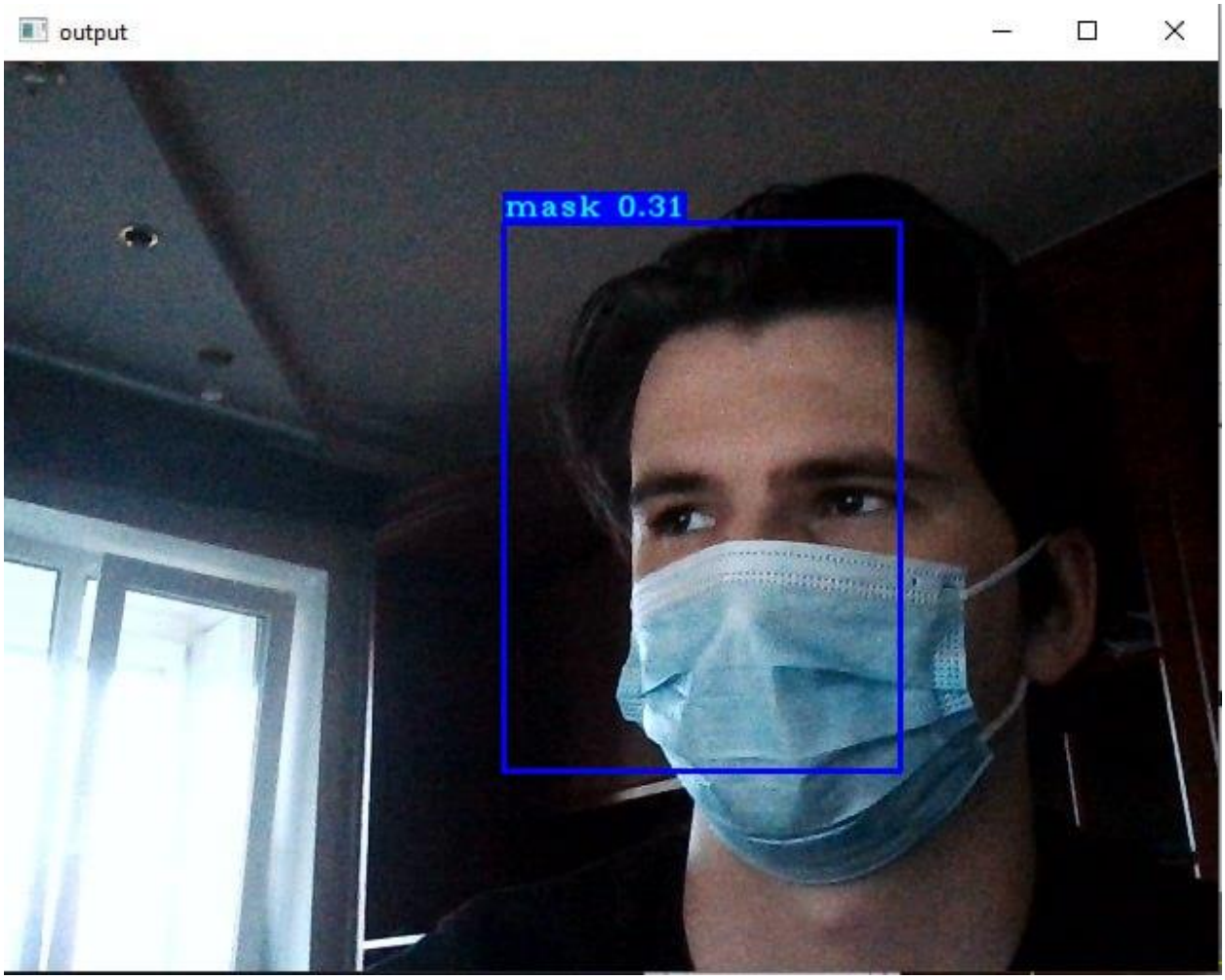

Рисунок 3.5 – Перегляд відео

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2023 р.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ НАЯВНОСТІ АКСЕСУАРІВ НА ОБЛИЧЧІ ЛЮДИНИ

Графічний матеріал

КПІ.ІП-9308.045490.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олена ХАЛУС

Нормоконтроль:
_____Максим ГОЛОВЧЕНКО

Виконавець:
_____ Ярослав ГРИГОРЕНКО

Київ – 2023

					КПІ.ІП-9308.045490.06.99.ССВ	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використань	Літера / Маса / Масштаб
Розробив		Григоренко Я. С.				
Перевірив		Халус О. А.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Головченко М. М.			Програмне забезпеченння для розпізнавання наявності маски на обличчі людини	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-93
Затвердив		Жаріков Е. В.				

					КПІ.ІП-9308.045490.07.99.ССП	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна послідовностей	Літера / Маса / Масштаб
Розробив		Григоренко Я. С.				
Перевірив		Халус О. А.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Головченко М. М.			Програмне забезпечення для розпізнавання наявності аксесуарів на обличчі людини	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-93
Затвердив		Жаріков Е. В.				

					КПІ.ІП-9308.045490.08.99.КЕ	
Зм.	Арк.	№ документа	Підпис	Дата	Креслення вигляду екранних форм	Літера / Маса / Масштаб
Розробив		Григоренко Я. С.				
Перевірив		Халус О. А.				Аркуш / Аркушів
Т. кон.						
Н. кон.		Головченко М. М.			Програмне забезпечення для розпізнавання наявності маски на обличчі людини	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-93
Затвердив		Жаріков Е. В.				