Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи

магістр

(освітньо-кваліфікаційний рівень)

на тему: «Розробка методів та програмного додатку розпізнавання обличчя людини»

Виконав: студент II курсу
групи 1ПІ-19м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Штокал С.С.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Хошаба О.М.

Рецензент: д.т.н., проф. каф.КНВасілевський О.М.

ВНТУ – 2021

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Ступінь вищої освіти – магістр
Спеціальність 121 - «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
17 лютого 2021 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Штокалу Сергію Сергійовичу

1. Тема роботи: «Розробка методів та програмного додатку розпізнавання обличчя людини»
керівник роботи: Хошаба Олександр Мирославович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 9 березня 2021 року № 65.

2. Строк подання студентом роботи1 червня 2021 року.

3. Вихідні дані до роботи: Метод розпізнавання обличчяза рахунок розпізнавання рис верхньої частини обличчя, інтерфейс для клієнтських додатків, технології та програмні засоби для розробки серверного веб-додатку(ASP.NET core, PyQT5 Python з використанням QT builder та PyQT tools), проєктування архітектури бази даних (SQLServer) та вибір технологій і програмних засобів для розробки клієнтських додатків для веб та десктоп(Angular на JavaScript, PyQT5 на Python).

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану питання та постановка задач дослідження; розробка методів імоделей системи; програмна реалізація системи розпізнавання обличчя людини; тестування програми; економічна частина; висновки; перелік посилань; додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):
актуальність теми; мета, об'єкт та предмет дослідження; основні задачі; загальна структурна схема ІТ-системи; дизайн інтерфейсу клієнтської частини та панелі адміністратора; програмні засоби розробки системи розпізнавання обличчя людини зклієнтським додатком; результати тестування; висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Хошаба О. М., к.т.н, доцент кафедри ПЗ	18.02.2021	31.05.2021
5	Бальзан М.В., к.е.н., доцент кафедри ЕПВМ	18.02.2021	31.05.2021

7. Дата видачі завдання 18 лютого 2021 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строквиконання етапів роботи	Примітка
1	Аналіз, вибір та обґрунтування актуальності розробки	20.02.2021– 08.02.21	Вик.
2	Аналіз існуючих аналогів	01.03.21 – 12.03.21	Вик.
3	Розробка методів імоделей системи розпізнавання обличчя людини	13.03.21 – 22.03.21	Вик.
4	Розробка макетів графічного інтерфейсу	23.03.21 – 31.03.21	Вик.
5	Програмна реалізація системи	01.04.21 – 25.04.21	Вик.
6	Тестування роботи системи	26.04.21 – 30.04.21	Вик.
7	Економічна частина	01.05.21 – 16.05.21	Вик.
8	Оформлення матеріалів до захисту МКР	17.05.21– 31.05.21	Вик.

Студент **Штокал С. С.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Хошаба О. М.**
(підпис) (прізвище та ініціали)

Рецензент магістерської _____ **д.т.н., проф. каф. КН Василевський О.М.**
кваліфікаційної роботи(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі розроблено функціонал системи для розпізнавання обличчя людини. Удосконалено метод розпізнавання покритого маскою обличчя за рахунок додаткового розпізнавання рис верхньої частини обличчя, що забезпечує точність розпізнавання в умовах неповної видимості обличчя.

Серед головного функціоналу було реалізовано розпізнавання обличчя на відео, ідентифікація особи за зображенням, додавання до бази розпізнаних облич та перевірка наявності облич у базі. Реалізовано два клієнтські додатки: веб та десктоп, розроблено моделі системи, дизайн інтерфейсу, а також спроектовано архітектуру бази даних.

Для запуску системи було налаштовано серверне оточення для розміщення бекенду системи та бази даних. Серверначастина містить API для роботи зклієнтськими додатками. Програмну реалізацію системи було створено засобами фреймворків ASP.NET та PyQt, звикористанням мов програмування C#, Python та JavaScript відповідно до модулів системи. Використано базу даних SQLServer та інтегровані середовища розробки Microsoft Visual Studio, Pycharm, а також для розробки десктопної аплікації PyQT5 Python.

ANNOTATION

In the master's thesis functionality of the system for face recognition is developed. The method of recognition of the face covered by a mask is improved, due to recognition of features of the upper part of the face that provides accuracy of recognition in the conditions of incomplete visibility of the face.

Among the main features, face recognition on video, identification of a person by an image, adding recognized faces to the database and checking the presence of faces in the database were implemented. Two client applications were implemented: web and desktop, system models, interface design were developed, and database architecture was designed.

To start the system, a server environment was configured to accommodate the system and database backend. The server part contains an API for working with client applications. The software implementation of the system was created using the ASP.NET and PyQt frameworks, using the C #, Python and JavaScript programming languages according to the system modules. The SQLServer database and integrated development environments of Microsoft Visual Studio, Pycharm, as well as for the development of the desktop application PyQT5 Python were used.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження.Зовсім недавно маски були лише атрибутом лікарів іжителів забруднених промисловістю міст Китаю. Але тепер їх носять у всьому світі.Саме такі зміни у світі принесли неабиякий вплив на системи розпізнавання обличчя, адже розпізнати конкретну людину, коли вона у масці іколи без маски– це різні за складністю завдання. Раніше детектори були навчені розпізнавати звичайні обличчя, які могли дещо перекриватися окулярами, шарфом або зачіскою. Проте на початку пандемії COVID-19 до 80-90% людей з'явилися перед камерою в масці [1]. Це стало для систем відео моніторингу проблемою.Тому актуальною є розробка сервісу для розпізнавання обличчя людини в сучасних умовах.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася відповідно до плану науково-дослідних робіт кафедри програмного забезпечення.

Мета та завдання дослідження:

Метою роботи є підвищення ефективності розпізнавання обличчя людини в масці за рахунок методів розпізнавання рис верхньої частини обличчя, що забезпечує точність розпізнавання в умовах неповної видимості обличчя.

У відповідності до поставленої мети потрібно виконати такі **завдання**:

- розробити метод виявлення та обробки рис верхньої частини обличчя;
- розробити метод розпізнавання обличчя, що поєднує функціонал методів розпізнавання, рис верхньої частини обличчя та рис всього обличчя;
- розробити модель системи розпізнавання обличчя людини;
- розробити дизайн інтерфейсу клієнтської частини додатку;
- розробити програмну реалізацію системи розпізнавання обличчя;
- спроектувати архітектуру бази даних веб-системи;
- розробити серверну частину додатку та API;
- зібрати базу відео до тестування системи;
- протестувати систему розпізнавання обличчя людини.

Об’єктом дослідження є процес розпізнавання обличчя людини зчастково закритими його частинами.

Предметом дослідження є програмні засоби для розпізнавання обличчя людини у звичайних умовах, а також зчастково закритими його частинами.

Методи дослідження. У процесі дослідження використовувались такі методи:

- методи розробки графічного веб-інтерфейсу додатку для розпізнавання обличчя людини;
- методи роботи збазами даних для побудови бази даних системи розпізнавання обличчя;
- методи побудови компонентів програми та їхніх взаємозв’язків для створення системи розпізнавання обличчя;
- методи проектування програмного забезпечення для виконання розробки програмного продукту.

Наукова новизна одержаних результатів:

1. Подальшого розвитку дістав метод виявлення та обробки рис верхньої частини обличчя на зображенню для подальшого збереження їх в базі даних, який, на відміну від існуючих, придатний для обробки облич, що частково покриті маскою, і орієнтований на захоплення рис верхньої частини обличчя, що забезпечує подальшу можливість розпізнавання.
2. Подальшого розвитку дістав комплексний метод розпізнавання обличчя, який, на відміну від існуючих, поєднує функціонал методів розпізнавання, рис верхньої частини обличчя та рис всього обличчя, що дозволяє підвищити точність розпізнавання обличчя з частково закритою частиною.
3. Подальшого розвитку дістали моделі системи розпізнавання обличчя, які, на відміну від існуючих, орієнтовані на реалізацію методу розпізнавання верхньої частини обличчя, що дозволяє отримувати високу точність розпізнавання в умовах часткового покриття обличчя маскою.

Практична цінність отриманих результатів. Практична цінність результатів полягає в тому, що на основі отриманих у магістерській кваліфікаційній роботі теоретичних положень розроблено програмні засоби для розпізнавання обличчя людини зчастково закритим обличчям у веб середовищі та в десктопній реалізації, що розширює перспективність практичного використання створеної системи.

Особистий внесок здобувача. У магістерській кваліфікаційній роботі усі результати дослідження здобуті автором даної роботи самостійно. У роботі[2], опублікованій у співавторстві, автору належить розроблена модель системи розпізнавання обличчя.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення й результати досліджень представлені наXII Міжнародної науково-технічної конференції "Інформаційно-комп'ютерні технології – 2021 (ІКТ-2021)". м. Житомир, 01-03 квітня 2021 р.

Публікації. За тематикою дослідження опублікована 1 наукова публікація [2] у матеріалах міжнародної конференції "Інформаційно-комп'ютерні технології – 2021 (ІКТ-2021)".

Структура та обсяг роботи.Магістерська кваліфікаційна робота містить вступ, п’ять розділів, висновки, список літератури, що містить 30 найменувань, та 6 додатків. Робота містить 26 ілюстрацій та 5 таблиць. Ілюстративний матеріал до роботи подано у додатку Е.

1 АНАЛІЗ РОЗВИТКУ СИСТЕМ РОЗПІЗНАННЯ ОБЛИЧЧЯ ТА ПОСТАНОВКА ЗАДАЧ РОБОТИ

1.1 Аналіз сервісів для розпізнання обличчя

Хоча розпізнавання обличчя існує в тій чи іншій формі з 1960-х років, останні технологічні розробки призвели до широкого розповсюдження цієї технології [3]. Ця технологія більше не сприймається як щось із науково-фантастичних фільмів. Наприклад, з виходом iPhone X мільйони людей тепер буквально мають технологію розпізнавання обличчя на долонях [4], захищаючи свої дані та особисту інформацію. Наприклад, контроль доступу до мобільного телефону може бути найбільш впізнаваним способом використання розпізнавання обличчя, він застосовується для широкого кола випадків використання, включаючи запобігання злочинам, захист подій та роблячи авіаперевезення більш зручними.

Технології розпізнавання вже активно використовуються в різних напрямках. Розглянемо кілька застосувань цієї технології.

Як вже було згадано, перше – це безпека пристрою. Деякі додатки використовують розпізнавання осіб для захисту особистих даних. Навіть безпечний пароль не може повністю захистити акаунти і інформацію від досвідчених хакерів, тому люди вирішили вдатися до технологій розпізнавання облич. Ці додатки вимагають показати їм обличчя, щоб розблокувати смартфон або отримати доступ до особистих даних.

Ще одним досить не типовим застосуванням є - виявлення генетичних порушень за допомогою розпізнавання. Існують спеціальні медичні програми, такі як Face2Gene [5] і DeepGestalt[6], які використовують розпізнавання облич для виявлення генетичних порушень. Вони аналізують обличчя і порівнюють їх з базою даних осіб тих людей, у яких є різні порушення.

Запобігання магазинних крадіжок. Багато магазинів оснащені системами розпізнавання осіб, які виділяють людей в якості загрози, якщо вони щось крали в магазинах. Така система може ідентифікувати магазинного злодія і повідомити власника магазину про його минулі витівки, навіть якщо такий злодій ніколи не бував в даному магазині раніше. Хоч така система може надавати значні вигоди для власників магазинів, але часто ефективність таких систем ставиться під сумнів. Якщо невинна людина буде позначена як злодій, то це може досить негативно вплинути на його життя.

Те ж саме стосується розпізнавання облич під час купівлі алкоголю в магазині. Тут іде мова про те, що деякі продуктові магазин і бари в Великобританії використовують розпізнавання облич, щоб визначити, чи достатньо років покупцеві, щоб мати право покупки алкоголь. Продуктові магазини дозволяють покупцям використовувати систему самоперевірки без необхідності в додатковому співробітника, перевіряючи паспорта. Якщо система вважатиме, що клієнту менше 25 років, то він повинен буде пред'явити паспорт для перевірки.

Безпека в школах. Розпізнавання облич починають впроваджувати в школах. Одна школа в Швеції використовує FaceRecognitionTechnology для перевірки відвідуваності на уроках [7]. Школи в США, особливо в Нью-Йорку [8], починають тестувати використання технологій розпізнавання осіб в якості «системи раннього оповіщення» проти загроз з боку таких осіб, як сексуальні маніяки. Технологія також може розпізнавати 10 видів зброї для запобігання актів насильства в школах.

Використання в авіакомпаніях. Такі авіакомпанії як Delta і JetBluevикористовують розпізнавання осіб для ідентифікації пасажирів [9]. Біометричний сканування особи є необов'язковим, але дозволяє пасажирам використовувати свої обличчя в якості квитка, економлячи час і скорочуючи витрати на перевірку квитків.

Додатки, розважального змісту, зазвичай мобільні аплікації. Наприклад, віковий фільтр FaceApp [10], який використовує розпізнавання осіб для

старіння вашого обличчя, набрав обертів у світі соцмереж. На жаль, існують побоювання, що зібрані ним дані про осіб не захищені належним чином.

1.2 Порівняння алгоритмів розпізнавання обличчя

Як відомо, розпізнавання обличчяпрацює шляхом ідентифікації на обличчі людини кількох ключових точок, а їх сполучення формують унікальне «графічне» зображення. Ці ключові точки зазвичай знаходяться навколо очей, носа і губ. Щоб система могла працювати при закритій нижній половині особи дослідники розташували більше ключових точок навколо очей і носа. Таким чином працюють більшість популярних алгоритмів.

Сьогодні широкого розвитку набув алгоритм AAM [11]– activeappearancemodels– активні моделі зовнішнього вигляду. Моделі мають інтегровану статистичну модель, що поєднує форму варіацій з варіаціями зовнішнього вигляду. Структура AMM залежить від піраміди гауссівского зображення, яка забезпечує швидкий і надійний підхід з декількома дозволами для ідентифікації. Складність цього підходу полягає в тому, щоб зрозуміти івивчити різницю між реальним інтервалом зміни даних і змінюється тільки через систематичні і безсистемних спотворень. Метод AMM є одним з найстаріших статистичних методів зіставлення, який досяг 88% точності, який навчений на 100 зображеннях осіб, помічених вручну, для навчального набору і рівної кількості осіб для набору для тестування, включаючи очікування осіб. Алгоритм моделі Active Appearance має дві процедури: моделювання і підгонку. На етапі моделювання AMM розділяє об'єкт на дві частини, перша - це форма, яка являє собою вектор, встановлений шляхом з'єднання лицьових орієнтирів, друга частина - це текстура, яка є мірою пікселів, представлених щільністю квітів. Після того, як модель сформована, необхідно підігнати її під різні зображення, які життєво важливі для визначення найбільш реалістичних параметрів особи.

HMM [12]– прихована модель Маркова - це математичний підхід, що використовується для моделювання біологічних послідовностей. HMM ефективно використовувався з одновимірними даними та досяг важливих результатів у розпізнаванні руху та розпізнаванні голосу. Він також використовувався для виявлення та розпізнавання обличчя.

PCA – Principal component analysis [13]– аналіз основних компонентів - це статистичний підхід до зменшення кількості параметрів при розпізнаванні обличчя. PCA – це класична техніка вилучення ознак та подання даних, яка широко використовується в областях розпізнавання зразків та комп'ютерного зору, таких як розпізнавання облич". У PCA кожне зображення у навчальному наборі даних представляється як лінійна комбінація зважених власних векторів, що називаються власними поверхнями. Ця методика використовувала багатогранний метод аналізу даних для дослідницького аналізу даних.

LDA – лінійний дискримінантний аналіз [14] – це техніка, яка використовується для зменшення розмірів набору даних та збереження якомога більше даних. Він генерує ідеальну функцію лінійної дискримінації, яка відображає вхідні дані в місці класифікації, вхідні дані оброблятимуться через розсіяні матриці.

LDA забезпечує роздільність класів, формуючи область прийняття рішень між різними класами. LDA намагається максимізувати співвідношення дисперсії між класами та дисперсії всередині класу. Метод створює лінійне їх об'єднання, що призводить до вищих середніх відмінностей між класами. Підхід LDA успішно застосовується в декількох додатках, таких як ідентифікація зображень, розпізнавання зразків, класифікація даних, біоінформатика та ін.

Вісь Z в методі LDA перетворює вектори вищих розмірів для різних класів у систему характеристик нижчого розміру; у цьому діапазоні нижчі вектори поділяються на класи. Тобто зменшення d-вимірного (Rd) простору до h-вимірного (Rh) простору (де d> h), призводить до того, що розмір орієнтації Z буде дорівнювати добутку h*d. Інакше кажучи, метод LDA створює

діагональну вісь і відображає інформацію від обох функцій, таким чином зменшуючи відхилення та збільшуючи відстань між двома класами, як показано на рисунку1.1.

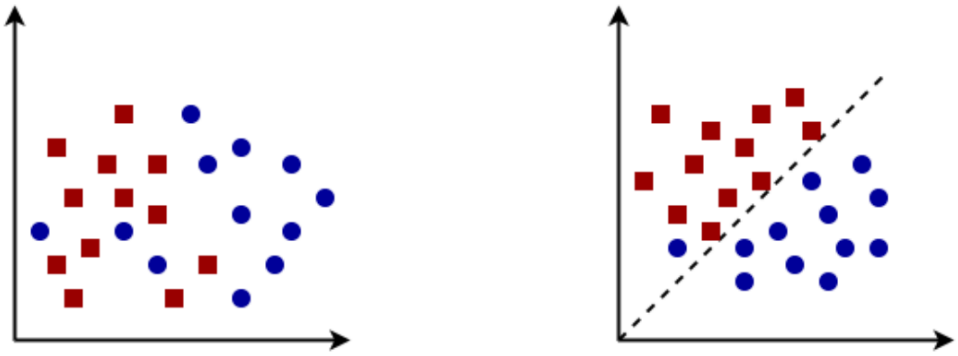


Рисунок 1.1- Вплив LDA на дані для розділення класів, враховуючи, що кожен колір є змінною

Таким чином, алгоритм LDA є прикладом моделі, що належить до галузі штучного інтелекту, яка дозволяє досягти високої точності під час процесу розпізнання та класифікації об’єктів. Саме тому було прийнято рішення використовувати саме його під час розробки власного програмного продукту.

1.3 Порівняльний аналіз аналогів

Існує досить багато реалізацій, направлених на вирішення проблеми розпізнавання обличчя людини. Розглянемо декілька найпоширеніших з них.

Animetrics Face Recognition[15] (рисунок 1.2) – сервіс представляє собою API, яке можна використовувати для пошуку людських облич, виявлення точок об’єктів, виправлення знімків під кутом та, зрештою, розпізнавання обличчя. Аплікація здатна розпізнати таку інформацію, як риси обличчя, включаючи вуха, ніс, брови, губи, підборіддя тощо. API розпізнавання облич Animetrics також виявляє стать. Серед унікальних можливостей доступна спеціальна функція, яка називається "SetPose", що дозволяє рендерити обличчя в бажаній

позі, яка відрізняється від захопленої. Тобто так, ніби фотографія зроблена з бажаного кута відносно камери.

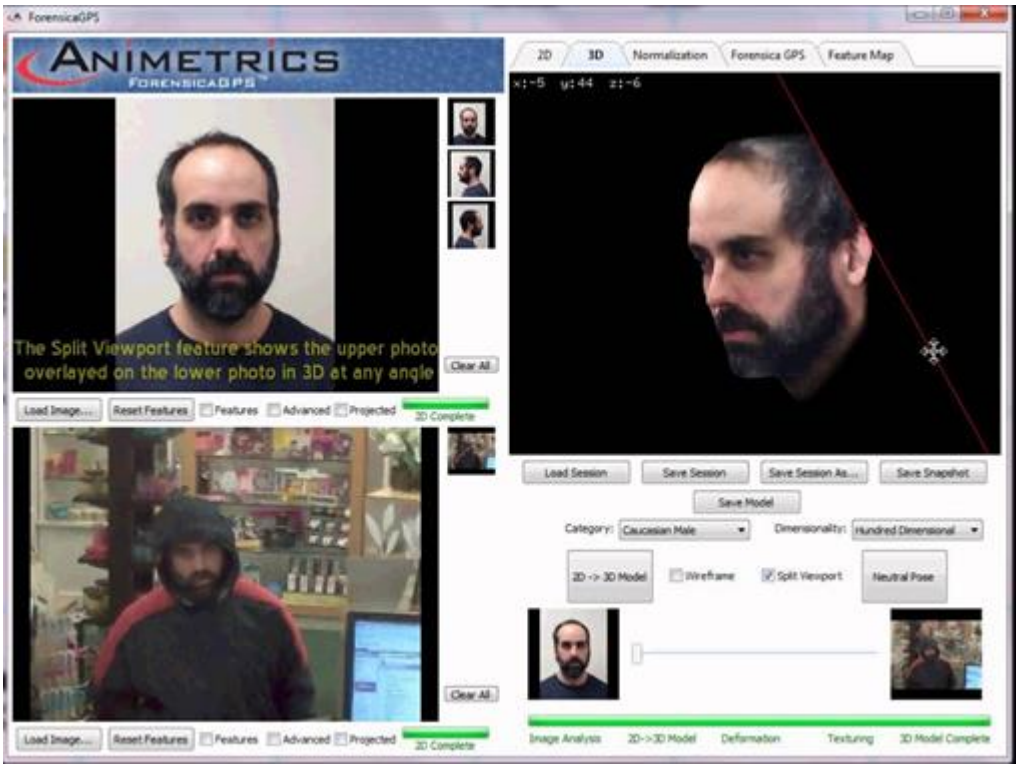


Рисунок 1.2 – ДодатокAnimetricsFaceRecognition

Виявлені риси обличчя можуть бути виправлені або модифіковані для поліпшення кінцевих результатів наступних етапів розпізнання. Наприклад, око на малюнку може бути прихованим або затемненим, система може в деякий спосіб модифікувати колір зображення, щоб підвищити точність розпізнання.

Серед недоліків можна назвати, такі, що система не справляється з поставленою задачею за умови покриття обличчя більш ніж на 50 %.

Luxand.cloud FaceRecognition[16] (рисунок 1.3) – сервіс, що дозволяє виявити та порівняти людські обличчя, визначите раніше позначених людей на зображеннях, розпізнати вік, стать та емоції на фото.

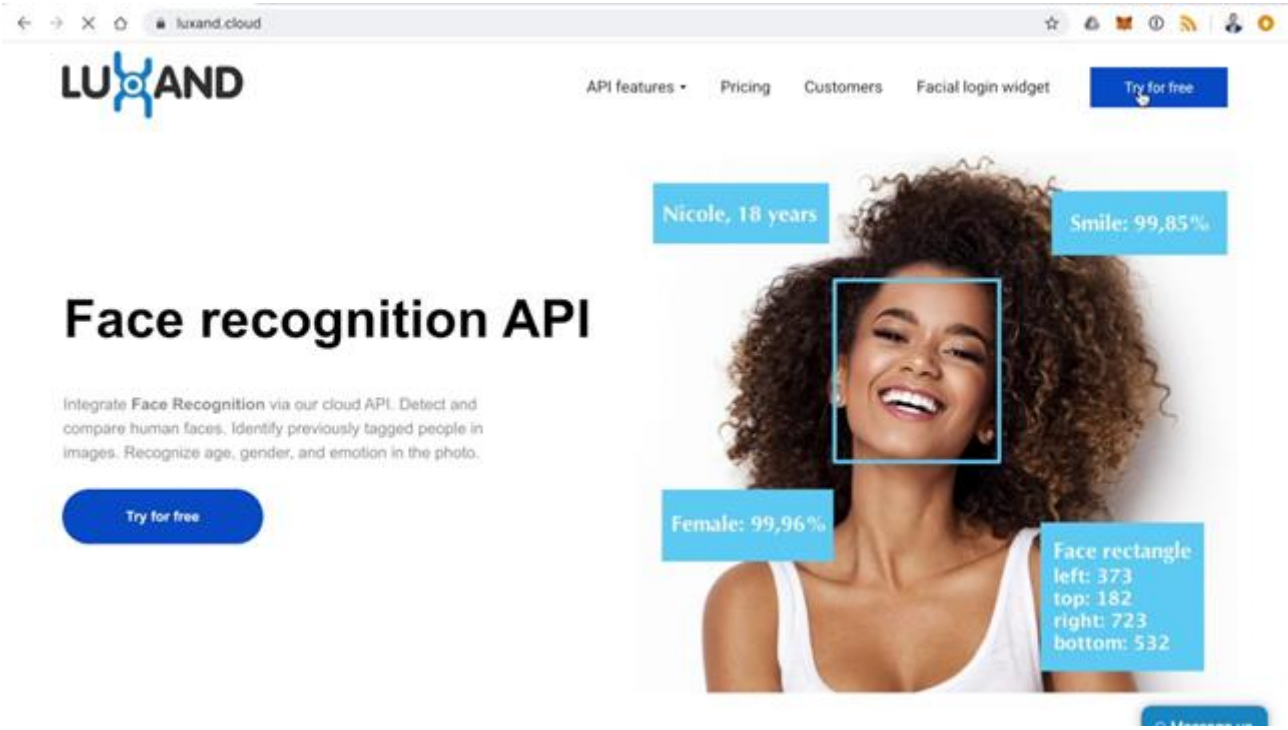


Рисунок 1.3– ПрограмаLuxand.cloud Face Recognition

Серед основних можливостей API можна визначити такі:

- пошук людей на конкретній фотографії;
- перевірка особи за обличчям;
- пошук конкретної людини в колекції фотографій;
- визначення статі, віку та емоцій на фото.

Додатково можна створити галерею для зберігання зображень у хмарі з розпізнаванням обличчя. До недоліків можна віднести зниження якості розпізнавання для зображень знедостатнім освітленням, а також з частковим перекриванням обличчя зачіскою, окулярами тощо.

Програмне забезпечення SenseTime[17] та однойменна компанія є одним із провідних постачальників штучного інтелекту у Китаї і світі (рис. 1.4). Програма побудована на алгоритмі, призначеному для зчитування 240 ключових точок особи навколо очей, рота і носа. Завдяки чому може бути знайдена відповідність, використовуючи тільки ті частини обличчя, які видно. Іншими словами, ключових точок навіть навколо очей може бути досить для складання унікального відбитка, нехай і часткового відбитка особи.

Рисунок 1.4– Програмне забезпечення SenseTime

Серед недоліків самі ж розробники відзначають неточності розпізнавання в умовах недостатньої освітленості та при поворотах обличчя, а також для облич, що, наприклад, покриті маскою, а також погано справляється з випадками, коли очі зариті окулярами, або у випадку, коли людина використовує штучну бороду і т.д. Широкого застосування програма досягла у офісних установах і використовуються переважно для розпізнавання співробітників компаній для обліку робочого часу.

1.4 Постановка задач роботи

Проаналізувавши подібні програмні рішення до сервісу магістерської кваліфікаційної роботи, було вирішено скласти список завдань, необхідних для створення власного програмного продукту.

Головною задачею роботи є створення системи, що дозволить розпізнавати людські обличчя в умовах часткового їх покриття медичними масками, шарфами тощо. Виконання цієї задачі передбачає:

– розробку методу розпізнавання обличчя, за рахунок розпізнавання рис верхньої частини обличчя;

- розробку моделі системи розпізнавання обличчя людини;
- розробку дизайну інтерфейсу для клієнтських додатків;
- програмну реалізацію системи розпізнавання обличчя людини;
- проектування архітектури бази даних веб-системи;
- налаштування серверного оточення для розміщення бекенду системи та бази даних;
- розробка серверної частини та програмних засобів системи, що містить API для роботи з клієнтськими додатками для веб та десктоп версії;

1.5 Висновки

Було розглянуто рольрозпізнавання обличчя людини на відео у сучасному світі. Досліджено види сервісів для розпізнавання, розглянуто їх основне призначення та функції. Розглянуто існуючі готові реалізації та проведено аналіз аналогів. В ході перегляду аналогів системи було визначено їх переваги та недоліки, зокрема точність розпізнавання за різних умов освітленості, наявності чи відсутності перешкод на обличчі, таких як окуляри чи медична маска. Було прийнято рішеннярозробити власний метод розпізнавання обличчя на основі розпізнавання рис верхньої частини обличчя. Визначено основні задачі розробки.

2 РОЗРОБКА МОДЕЛІІ МЕТОДІВРОБОТИ СИСТЕМИ

2.1 Аналіз інформаційного забезпечення системи

Однимізавдань під час реалізації системи є вибір технологій, які на високому рівні зможуть задовольнити технічні, а також функціональні вимоги створюваних модулів системи.

Система складається з серверної частини, а також двох клієнтських додатків для веб та десктоп версій. Клієнт-серверна архітектура – збірне поняття, що складається з двох взаємодоповнюючих компонентів: сервера і, власне, клієнта.

Клієнт – локальний комп'ютер на стороні віртуального користувача, який виконує відправку запиту до сервера для можливості надання даних або виконання певної групи системних дій.

Сервер – дуже потужний комп'ютер або спеціальне системне обладнання, яке призначається для вирішення певного кола завдань по процесу виконання програмних кодів. Він виконує роботи сервісного обслуговування за клієнтськими запитами, надає користувачам доступ до певних системних ресурсів, зберігає дані або БД.

Особливості такої моделі полягають в тому, що користувач відправляє певний запит на сервер, де той системно обробляється і кінцевий результат відсилається клієнту. У можливості сервера входить одночасне обслуговування відразу декількох клієнтів.

Якщо одночасно надходить більше одного запиту, то такі запити встановлюються в певну чергу і сервером виконуються по черзі. Часом запити можуть мати свої власні пріоритети. Частина запитів з більш високими пріоритетами будуть постійно виконуватися в першочерговому порядку.

Параметри, які можуть реалізуватися на стороні сервера:

- зберігання, захист ідоступ до даних;

- робота зклієнтськими запитами;
- процес відправки відповіді клієнту.

Параметри, які можуть реалізуватися на стороні клієнта:

- середовище з надання графічного інтерфейсу;
- формулювання запиту до сервера і його подальша відправка.

Отримання підсумків запиту івідправка додаткової групи команд (запити на додавання, оновлення інформації, видалення групи даних).

Архітектура системи клієнт-сервер формулює принципи віртуального спілкування між локальними комп'ютерами, а всі правила і принципи взаємодії знаходяться всередині протоколу.

Мережевий протокол – це особливий набір правил, на підставі якого виконується точна взаємодія між комп'ютерами усередині віртуальної мережі.

Клієнт спілкується з сервером за допомогою протоколу HTTP. Контролер веб-API схожий на контролер ASP.NET MVC. Він обробляє вхідні HTTP-запити та надсилає відповідь абоненту.

Зазвичайу контролері реалізовано такі методи як GET, POST, PUT та DELETE. Кожен з методів дозволяє обмінюватись інформацією між клієнтською та серверною частинами.

Найзручнішим іфункціональним, на мій погляд, форматом обміну даними є JSON, але ніхто не забороняє використовувати XML, YAML або будь-який інший формат, що дозволяє зберігати серіалізовані об'єкти без втрати типу даних.

За бажанням можна зробити в API підтримку декількох форматів введення / виведення. Досить задіяти HTTP заголовок запиту для вказівки бажаного формату відповіді Accept і Content-Type для вказівки формату переданих в запиті даних.

Іншим популярним способом є додавання розширення до URL ресурсу, наприклад, GET /users.xml, але такий спосіб здається менш гнучким і красивим, хоча б тому, що ускладнює URL і вірний швидше для GET-запитів, ніж для всіх можливих операцій.

Web-API складних систем, що підтримують кілька користувальницьких ролей, часто вимагає поділу на частини, кожна з яких обслуговує свій спектр завдань.

2.2 Розробка загальної моделі системи

Загальнафункціональна структурнамодель IT-системи зображена на рисунку 2.1.

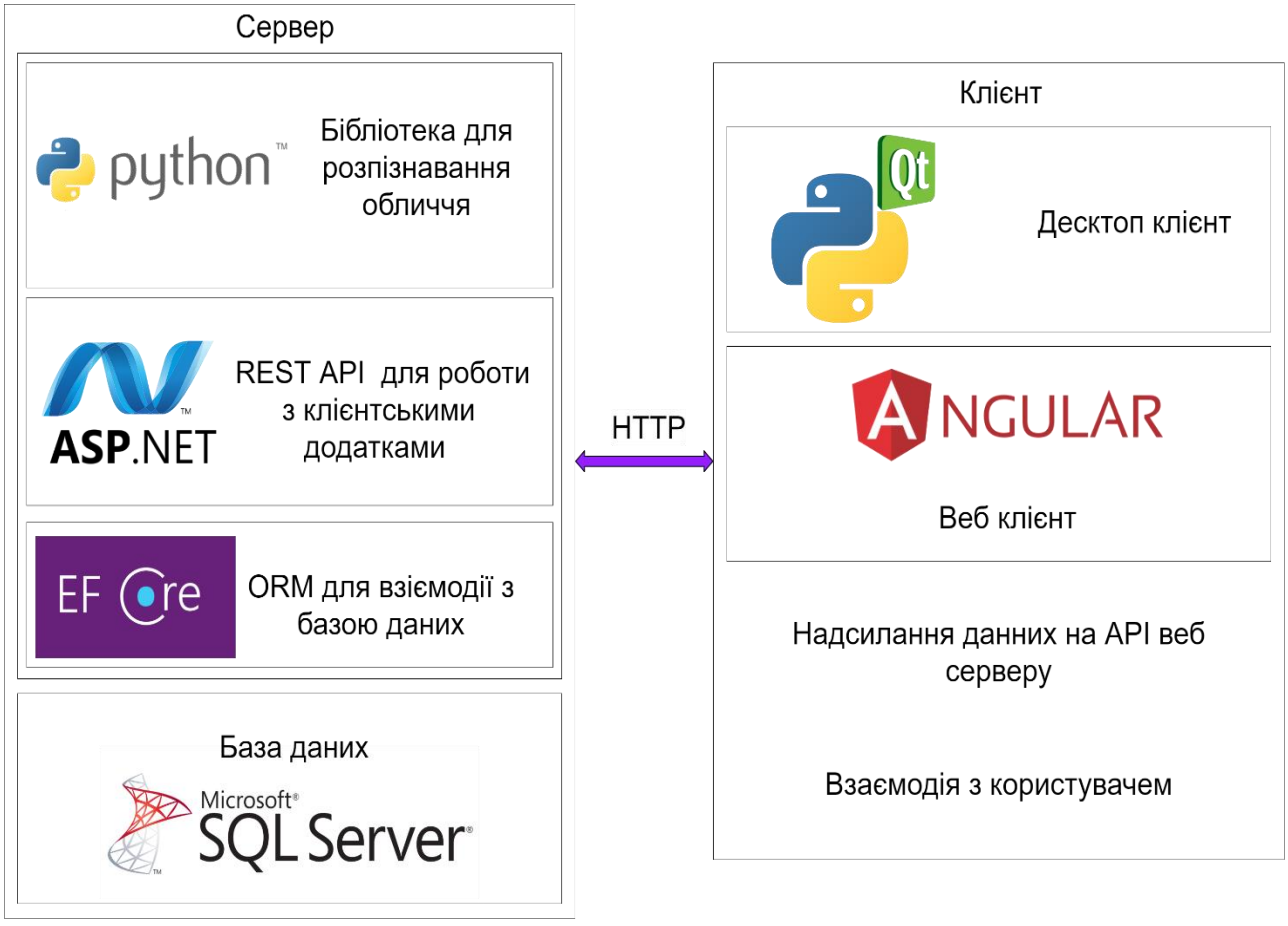


Рисунок 2.1– Загальна структурна модель IT-системи

Система розпізнання обличчя складається зосновних двох частин– Front end - клієнтська частина та backend – серверна частина.

Серверна частина складається з таких компонентів, як: база даних, API серверний клієнт і написаний власноруч модуль для розпізнання обличчя.

Клієнтська частина складається з 2 версій: версія для стаціонарного комп'ютера і для веб браузера. Для десктоп додатку використано Qt for Python (pyqt5-tools [18]) –PyQt – один з найпопулярніших прив’язок Python для крос-платформної розробки графічних аплікацій.

Для вебклієнтавикористано фреймворк Angular– це веб-фреймворк з відкритим кодом на основі TypeScript[19], яку очолив Angular Team[20] в Google та спільнотою приватних осіб і корпорацій. Angular повністю переписаний командою Google з AngularJS.

За допомогою клієнтської частини користувач взаємодіє з серверною частиною за допомогою протоколу HTTP або HTTPS та web API серверної частини. Додатково в декстопній версії є можливість роботи оффлайн, тому що всі бібліотеки для розпізнання можуть бути встановлені локально.

Серверний блок, у свою чергу, розділений на кілька основних компонентів, включаючи створення бази даних та підключення веб-програми, яка працює з нею. Серверна веб-програма має бізнес-логіку та API для спілкування з клієнтом.

Розглянемо детально серверну частину системи. Для обробки даних, які висилаються з клієнтської частини, створено REST API за допомогою фреймворку ASP.NET Core[21]– це платформа веб-додатків на базі сервера з відкритим кодом, призначена для веб-розробки і створення динамічних веб-сторінок. Вона був розроблений в Microsoft, щоб дозволити програмістам створювати динамічні веб-сайти, програми та сервіси. Після того, як API сервер отримує данні, а саме зображення з обличчям людини, він може викликати скрипт Python, за допомогою якого здійснюється розпізнання обличчя і виявлення рис обличчя для подальшого записування їх до бази даних.

Аплікація ASP.NET Core взаємодіє з сервером бази даних MSSQL Server за допомогою Entity Framework Core–це сучасний мапер об'єктівбаз даних (ORM– Object–relational mapping)[22] для. NET. Він підтримує запити LINQ,

відстеження змін, оновлення та міграції схем. EF Core працює з багатьма базами даних, включаючи базу даних MsSQL server (локальну та Azure[23]), SQLite[24], MySQL[25], PostgreSQL[26] та Azure Cosmos DB[27]. В базі даних зберігаються фотографії і закодовані риси обличчя кожної людини яка записана до системи.Загальна модельпроцесурозпізнання обличчя наведена на рисунку 2.2.

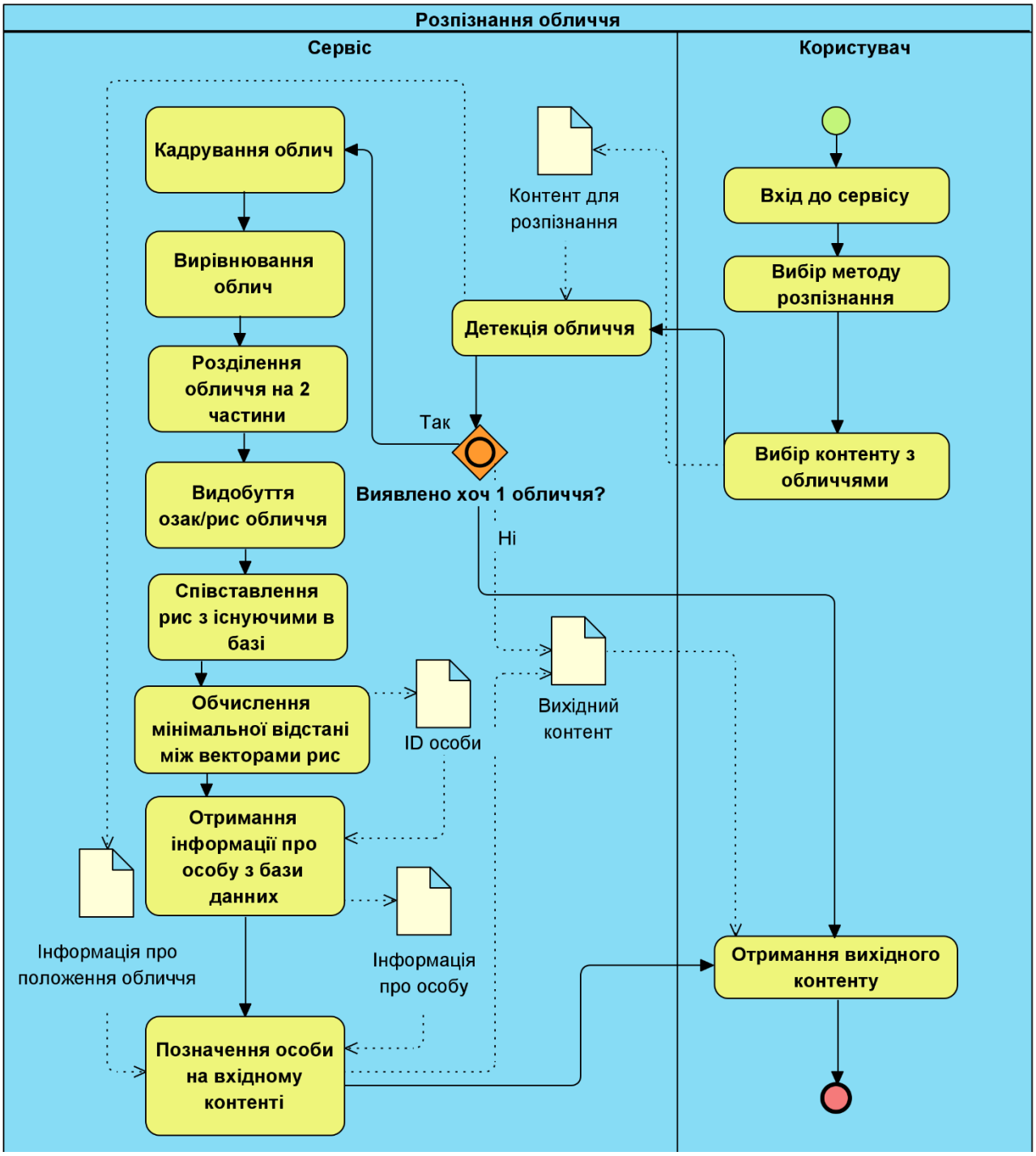


Рисунок 2.2 – Загальна модельпроцесурозпізнання обличчя

Аплікація, яка являє собою API сервер, може приймати зображення облич людей на вході, і повертати інформацію про цю людину (якщо вона раніше була записана до бази даних) і також координати розташування обличчя на зображенні.

Розпізнавання обличчя відбувається за допомогою бібліотеки з покращеним методом розпізнання, створеним власноруч.

Є 2 можливості використання бібліотеки: додавання нових облич до бази і розпізнання обличь на основі вже записаних до бази.

Можна також зберігати зображення і закодовані риси обличчя людини в 2 варіантах: для всього обличчя і тільки верхньої частини для подальшого кращого розпізнавання.

2.3 Розробка методу обробки рис обличчя на зображеннідляоблич, закритих маскою

Для того, щоб ефективно виявити риси обличчя людини, що знаходиться в масці,було вирішено розробити метод.

Метод полягає в тому, щоб розділити обличчя на 2 частини – верхню і нижню, і виявляти ту ж саму кількість рис, які виявляються для всього обличчя, тільки для верхньої його частини (рисунок 2.3).

Це дозволяє збільшити роздільну здатність розпізнавання видимої частини обличчя.

Загальна послідовність дій цього методу зображена на рисунку 2.4.

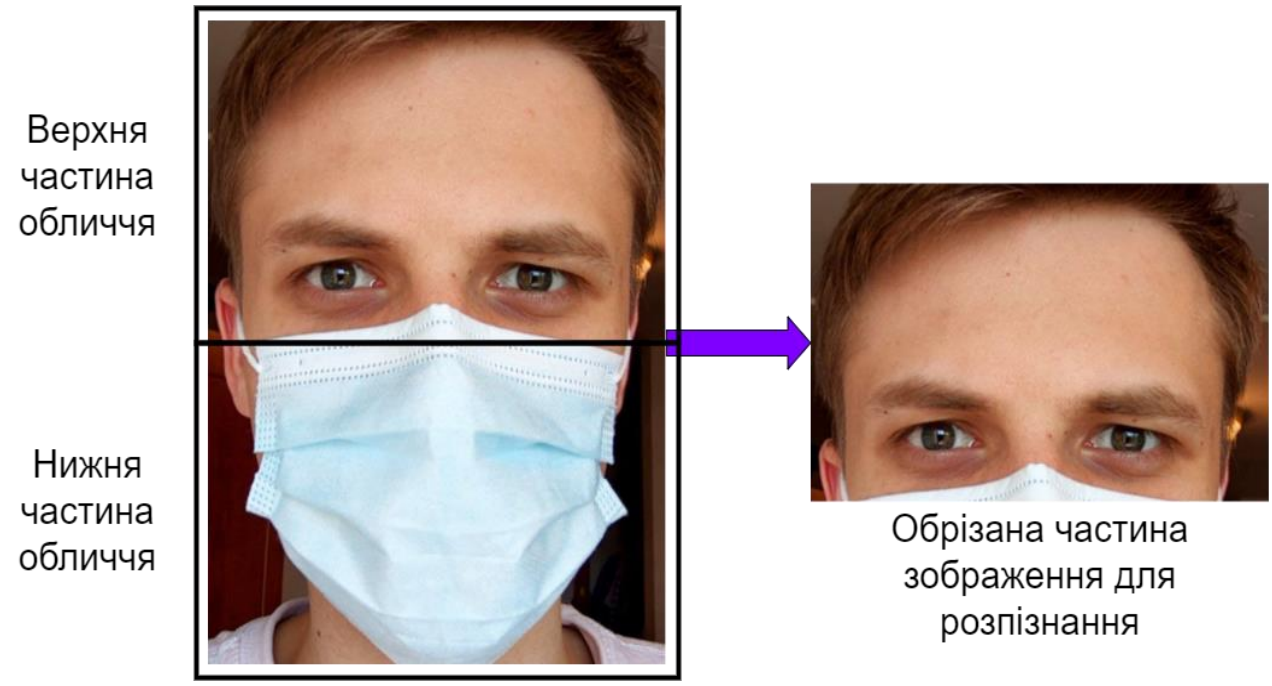


Рисунок 2.3– Ілюстраціявиділення верхньої частини обличчя в масці

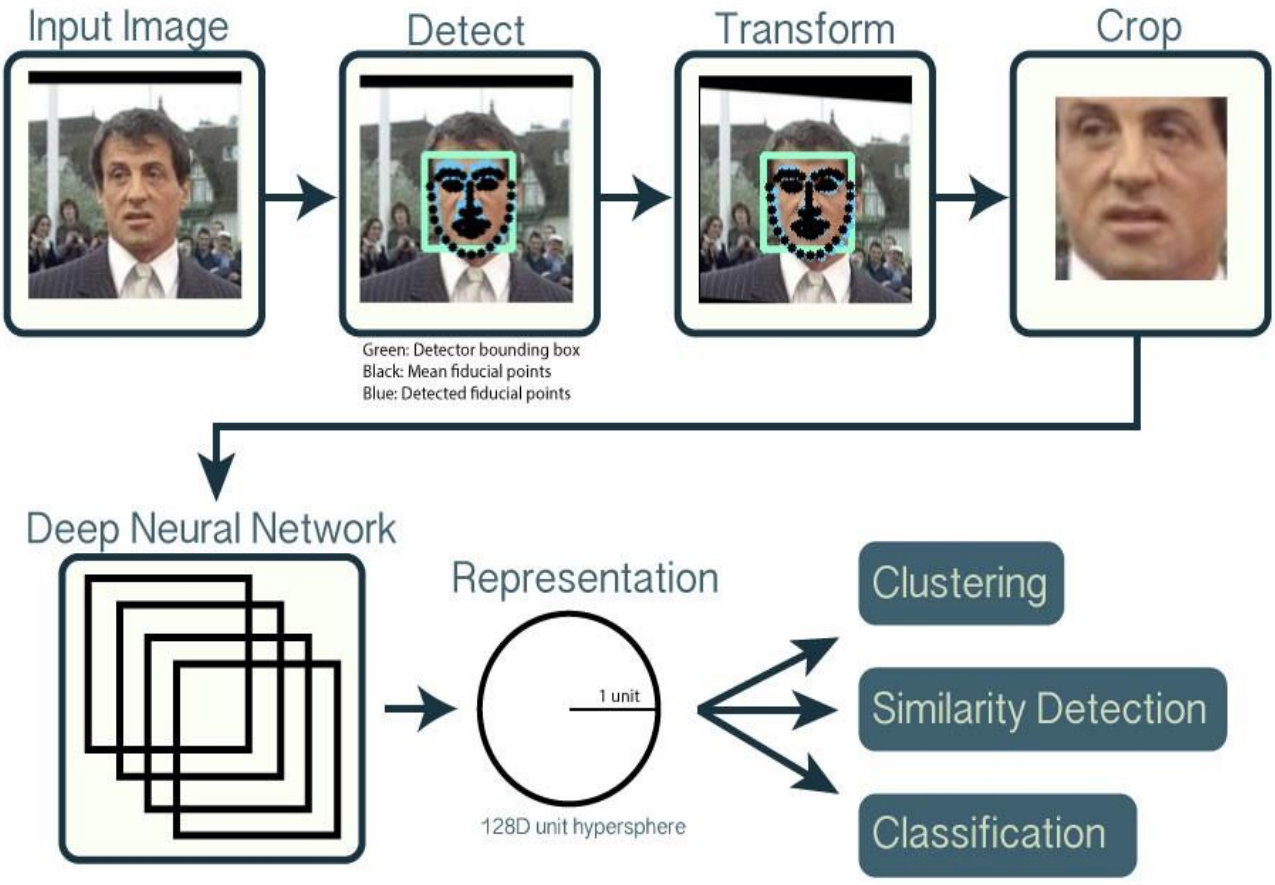


Рисунок 2.4–Процес розпізнання обличчя

1. Detect – детекція. На цьому кроці відбувається детекція обличчя на фотографії. Тобто визначається координати прямокутної області де знаходиться обличчя (рисунок 2.5). Уцій системі для детекції використовується нейрона мережа. Координати подаються у вигляді 2 точокP1та P2що описують прямокутну область на зображенню. Координати рахуються в кількості пікселей з лівого верхнього кута зображення. По горизонталі – вісь X, по вертикалі – вісь Y. Тому положення будь-якого обличчя на зображенні можна описати в такому вигляді:face_position=[P1=(x1,y1),P2=(x2,y2)].

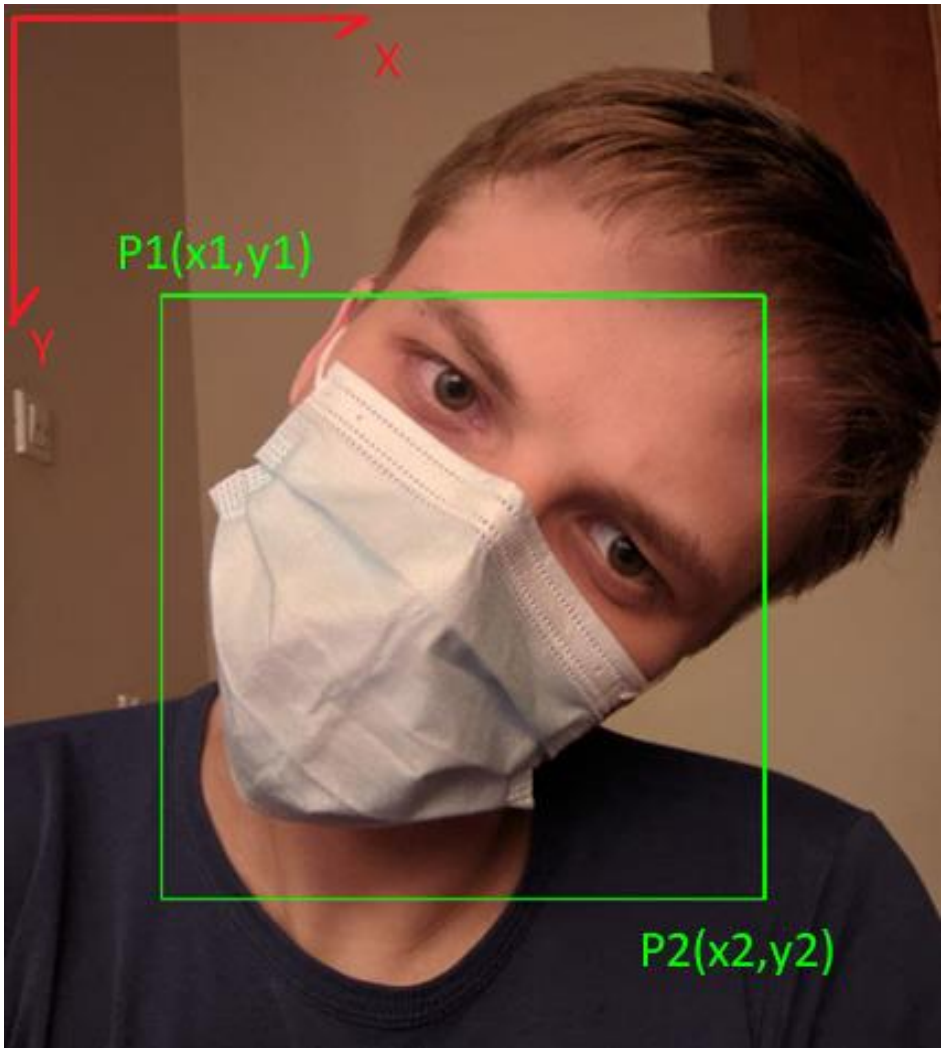


Рисунок 2.5– Виявлення обличчя

2. Transform – Трансформація або вирівнювання. Цей крок потрібен для того, щоб стандартизувати всі обличчя під один шаблон, для підвищення ефективності в подальшому розпізнанні.

Уцій системі для вирівнювання використовується метод з використанням 5 ключових точок обличчя(рисунок 2.7).По 2 точки припадає на кожне око і одна точна на кінчик носа.

У випадку коли людина носить маску, то вирівнювання відбувається за допомогою 4 точок очей (рисунок 2.6). Зображення деформується таким чином, щоб обличчя було ідеально перпендикулярне горизонту (рисунок 2.7).

3. Crop – Обрізання зображення щоб на ньому знаходилось тільки обличчя. Це можна побачити на рисунку 2.8.

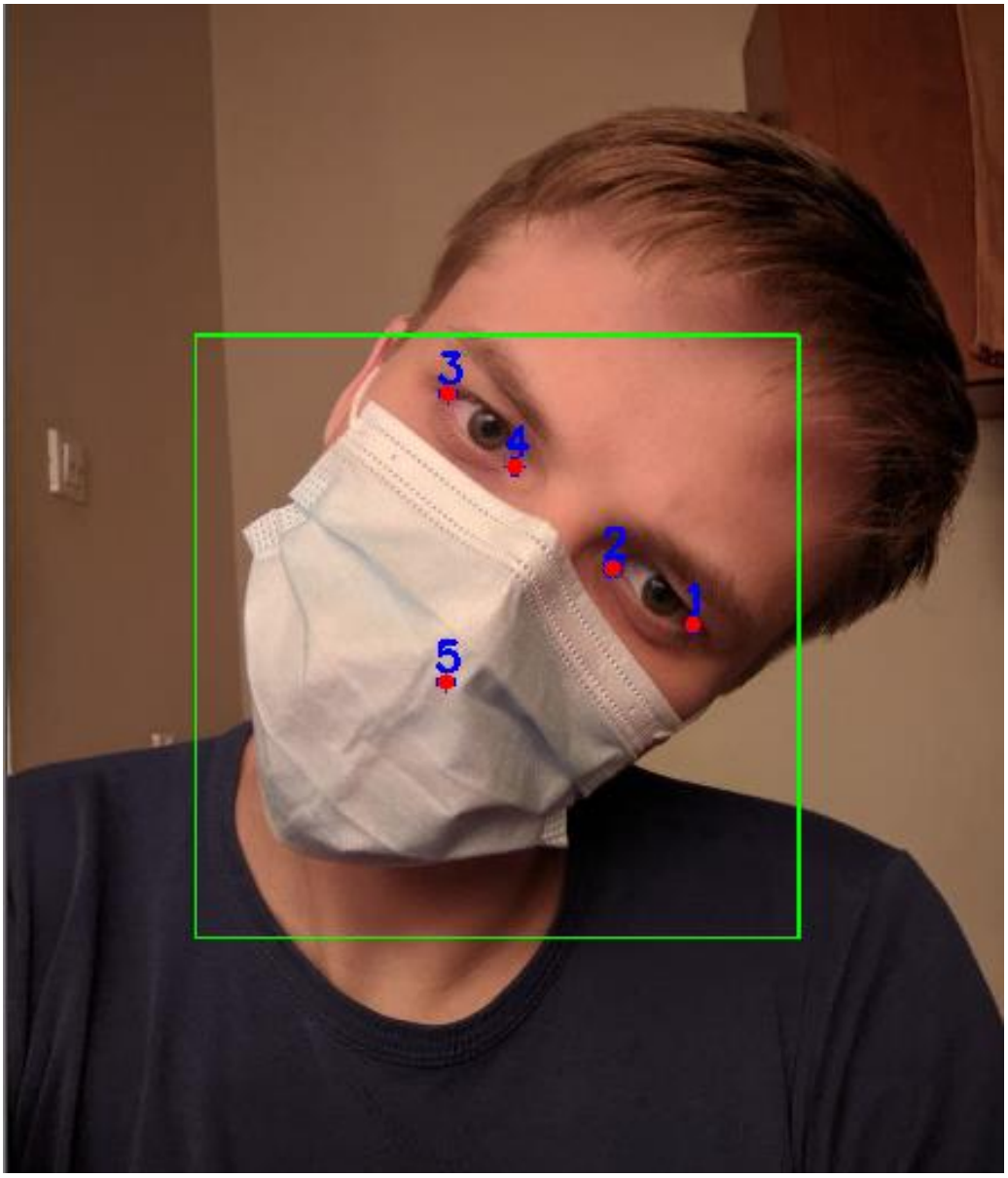


Рисунок 2.6–Виявлення ключових точок дляобличчя в масці

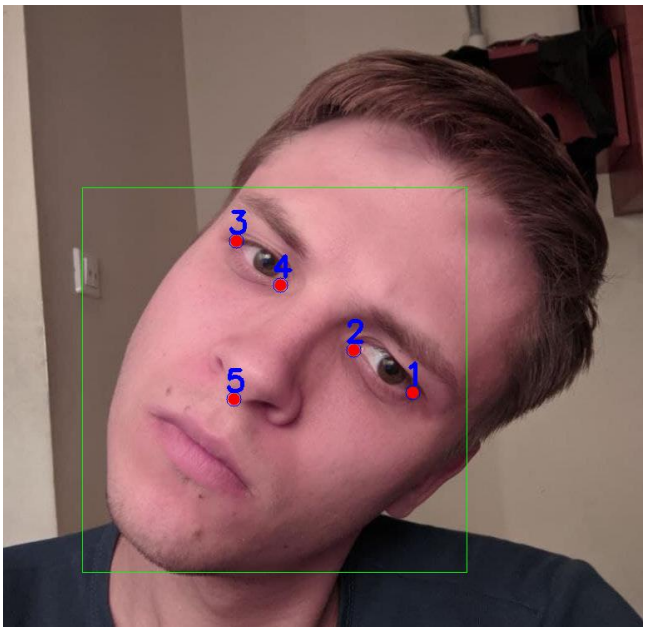


Рисунок 2.7– Виявлення ключових точок для обличчя без маски

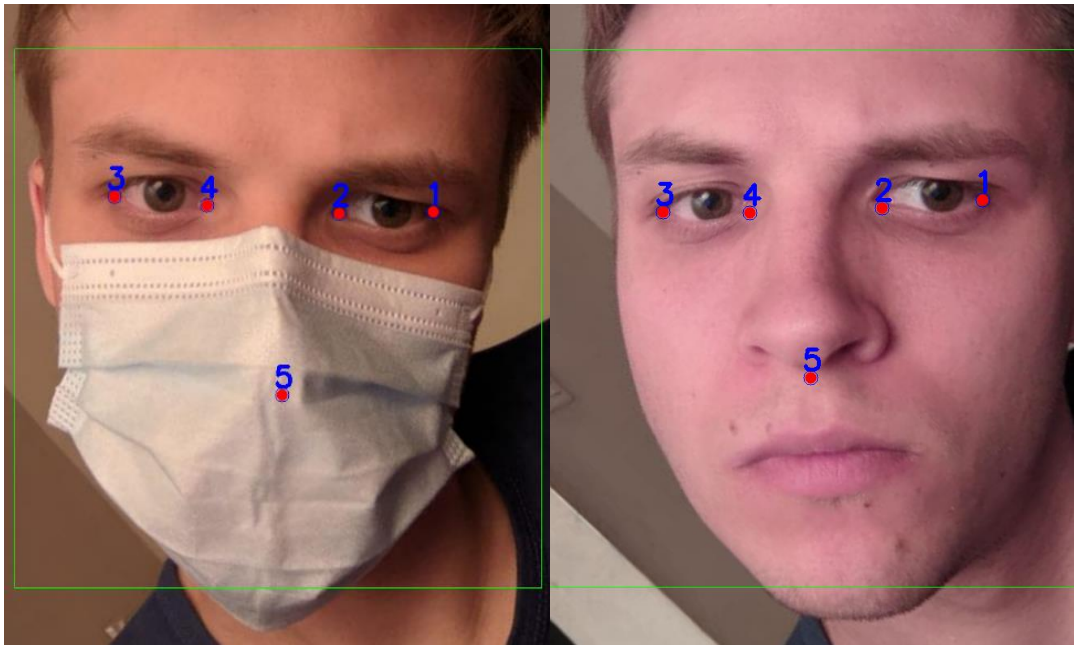


Рисунок 2.8– Обрізане зображення після вирівнювання на ньому обличчя.

Після цього кроку відбувається розділення обличчя на 2 частини, оскільки зображення вирівняно, то можна легко визначити його верхню частину розділивши зображення на 2 частини (рисунок 2.2). Після цього кроку отримуємо 2 зображення: вирівняне ціле обличчя і вирівняна верхня частина обличчя.

4. Deep Neural Network –на цьому кроці нейронна мережа обчислює 128-вимірний масив рис для обличчя (embeddings), а потім налаштовує ваги

мережі (за допомогою функції триплетних втрат). Модель, відповідальна за фактичну кількісну оцінку обличчя на зображенні, - це проект OpenFace, реалізація розпізнавання обличчя Python і Torch з глибоким навчанням. Ця реалізація походить від публікації [28].Для запропонованого методу потрібно 2 рази отримати риси: з всього обличчя та обрізаної його частини триманої з попереднього кроку. На виході маємо два 128 вимірні вектори.

5. Representation–З попереднього кроку маємовекторце і є репрезентація обличчя людини. В нашому випадку створюються 2 вектори – для всього обличчя і тільки для верхньої його частини (рисунок 2.9).Таким чином, мережа може якісно визначати риси обличчя та повертати дуже надійні та чіткі вбудовування, придатні для подальшого розпізнавання облич.

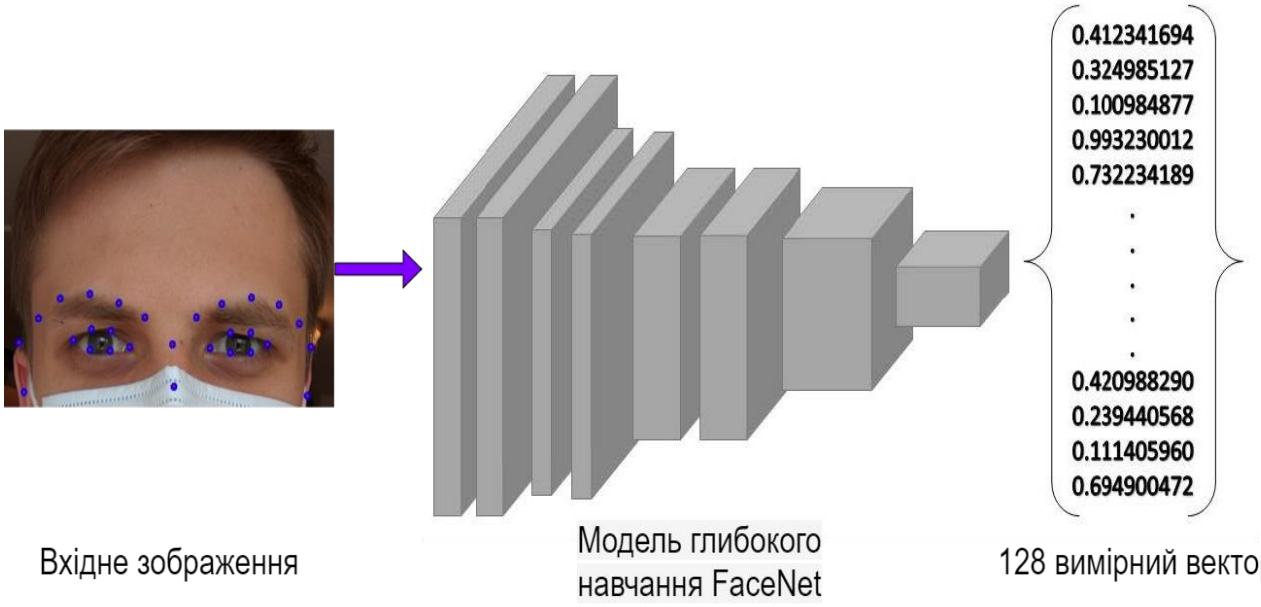


Рисунок 2.9– Перетворення рис верхньої частини обличчя на 128 вимірний вектор чисел

6. Після того як отримано вектор в якому закодовані риси, можна його порівнювати з іншими векторами для виявлення подібності облич (Similarity detection), можна також класифікувати ці вектори різними способами. На цьому

етапі і відбувається розпізнання[29].Метод розроблений для порівнювання облич описаний в наступному розділі.

2.4 Розробка комплексного методу розпізнавання обличчя

Для зручності введемо позначення UF (upper face) -верхня частина обличчя, FF (fullface) – ціле обличчя.AD (averagedistance) – середня відстань.

Завдяки попередньому методу покращеної обробки рис обличчя що знаходиться в масці маємо набір облич в базі даних. Для кожного обличчя є 2 записи рис для верхньої частини та всього обличчя.І до того ж кожна особа може мати декілька обличь записаних в базі даних. Велика вибірка облич для особи робиться для того щоб можна було якнайкраще ідентифікувати обличчя для різних поз і кутів під якими зняте обличчя.Для того щоб ідентифікувати особу запропоновано такий метод:

1. Отримати риси обличчя. На цьому кроці використовується метод детекції рис обличчя, що запропонований вище. На виході отримуємо 2 вектори рисобличчя (для всього і нижньої частини його) яке хочемо розпізнати.
2. Далі вибираємо з бази закодовані риси облич для кожної особи в 2 варіантах (всього обличчя і верхньої частини)
3. Беремо вектор рис всього обличчя яке хочемо розпізнати (з пункту1) далі для кожного вектору з бази отримуємо Евклідову відстань між нашим вектором рис який хочемо розпізнати. Ту ж саму відстань отримуємо для векторів верхньої частини обличчя. Відстань показує наскільки ці риси схожі. Відстань може бути від 0 до 1. Чим менша відстань тим більш подібні риси.
4. В наслідок чого отримаємо структуру, де для кожного обличчя в базі буде порахована відстань до обличчя,яке хочемо розпізнати. Для прикладу це зображено на таблиці 2.1 (3 перші колонки).
5. Розрахунок значення відстані з врахуванням відстані для UF та FF рис обличчя яке хочемо розпізнати з кожним обличчям в базі. Значення відстані обраховується за формулою 2.1

$$AD = \frac{FF_{distance} + k * UF_{distance}}{2} \#(2.1)$$

де k –коефіцієнт, який отриманий експериментальним шляхом і дорівнює 0.87,

FF_{di}- Евклідова відстань між векторами рис для всього обличчя,

UF_{di} - Евклідова відстань між векторами рис для верхньої частини обличчя. Для прикладу це показано в таблиці 2.1 в колонці 4.

Таблиця 2.1 - Приклад розрахунку для розпізнання обличчя

	Обличчя				
Номер особи	Все (FF)	Верхня частина (UF)	Значення відстані (AD)	Мінімальна відстань	Особа
1	0.45	0.2475	0.3326625	0.251125	3
1	0.85	0.85	0.79475		
1	0.35	0.175	0.251125		
2	0.65	0.975	0.749125	0.1500375	
2	0.8	0.4	0.574		
2	0.15	0.1725	0.1500375		
2	0.8	0.99	0.83065		
3	0.3	0.285	0.273975	0.073925	
3	0.8	0.99	0.83065		
3	0.1	0.055	0.073925		
3	0.5	0.5	0.4675		
3	0.85	0.99	0.85565		
3	1	0.99	0.93065		
3	0.15	0.1425	0.1369875		

6. Визначення найменшої відстані між серед всіх обличь певної особи. (Таблиця 2.1 колонка 5 зліва).

7. Визначення до якої особи найменша відстань. З прикладу на таблиці 2.1 маємо що до особи 3 найменша відстань, отже ті риси які ми захотіли розпізнати, а разом з ними і обличчя, належить 3-й особі.

8. Отримання з бази даних інформації про особу, яку отримали на кроці 6.

2.5 Висновки

У другому розділі магістерської кваліфікаційної роботи було проаналізовано інформаційне забезпечення системи. Розглянуто загальну функціональну структурну модель системи.Було розглянутоіснуючі алгоритми розпізнаваннята виявлено їх недоліки й переваги. Було запропоновано власний метод розпізнавання обличчята описано його роль у процесі розпізнавання.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Вибір мови програмування та фреймворку для розробки серверного додатку

Перш за все слід відзначити, що результатом магістерської кваліфікаційної роботи є система, а не самостійний програмний компонент. Щоб система працювала, потрібно реалізувати кілька модулів, які спільно реалізують повний цикл всієї системи в цілому. Серед них є серверні та клієнтські блоки.

Серверний додаток являє собою RESTful веб-вервіс. REST використовується для створення легких, підтримуваних і масштабованих веб-сервісів. Сервіс, побудований на REST архітектурі, називається RESTful-сервісом. REST використовує HTTP - базовий мережевий протокол.

Ключові складовими реалізації RESTful є:

Методи запитів. Вони кажуть, що хочемо зробити з ресурсом. Браузер використовує метод GET, щоб проінформувати віддалену сторону про те, що він хоче отримати дані. Крім GET є багато інших методів на кшталт POST, PUT і DELETE.

Заголовки запиту. Це додаткові інструкції, що посилаються разом із запитом. Вони можуть визначати тип необхідного ресурсу або подробиці авторизації.

Тіло запиту. Це дані, що відправляються разом із запитом. Дані зазвичай відправляються, коли виконується POST-запит до REST веб-сервісу. Найчастіше в POST-запиті клієнт говорить серверу, що він хоче додати на нього ресурс. Отже, тіло запиту містить детальну інформацію про ресурс, який необхідно додати на сервер.

Тіло відповіді. Це основна частина відповіді. Наприклад XML-документ з даними про розпізнавання.

Коди відповіді. Ці коди повертаються сервером разом з відповіддю. Наприклад, код 200 зазвичай означає, що при написанні відповіді не відбулося ніякої помилки.

Загальна структура серверної частини подана на рисунку 3.1.

Рисунок 3.1– Структура серверної частини додатку

У серверній частині реалізована основна логіка програми, вона реалізована на сервісному, а також бізнес рівні.

Серед основних функціональних можливостей можна виділити наступні:

- детекція обличчя на відео
- захоплення рухомої частини тулуба і розпізнавання обличчя
- занесення до бази нових розпізнаних облич
- ідентифікація особи, використовуючи базу облич

Одним із компонентів серверної частини є програмна реалізація побудови бази облич (додаток Б).

3.2 Розробка бази даних

Під час вибору бази даних було враховано те, що система потребує підтримки великої кількості важких операцій. з цією метою було прийнято рішення використовувати реляційну базу даних, яка дає можливість досить зручно провести аналогію між моделями системи і відповідними таблицями.

Для реалізації системи було обрано базу даних Microsoft SQL Server, який є досить сучасною платформою для збереження та обробки даних. Microsoft SQL Server має ряд переваг. Розглянемо деякі з них, щоб довести ефективність використання саме цієї бази розробляючи власний програмний продукт. За останні 10 років SQL Server пройшов шлях від рішення для невеликих і середніх СУБД до потужної платформи даних рівня підприємства, розрахованої на критичні бізнес-додатки по надійності і відмовостійкості. з кожним новим релізом SQL Server все більше має право називатися єдиним центром управління всіма даними. SQL Server враховує всі сучасні вимоги по роботі з даними різних форматів і з різноманітних джерел і стає природним вибором для побудови платформи інтеграції, управління та аналізу будь-яких даних.

SQL Server спрощує розгортання, передачу та інтеграцію великих даних

Рішення для обробки великих даних на основі Kubernetes, вбудоване в SQL Server, дозволяє легко розгорнути кластер великих даних і працювати з ним. Kubernetes забезпечує розгортання сховищ HDFS, реляційного модуля SQL Server і засобів аналітики Spark у вигляді контейнерів в рамках одного зручного пакету.

Сьогоднішні обсяги даних роблять нерозумним і невигідним конвертацію всіх доступних даних в реляційні таблиці для зберігання в СУБД. Ще 2 роки тому Microsoft представила технологію PolyBase, що дозволяє примірнику SQL Server обробляти запити Transact-SQL, які звертаються до даних Hadoop і об'єднувати дані з Hadoop і SQL Server. У SQL Server зовнішня таблиця або зовнішнє джерело даних забезпечує з'єднання з Hadoop, віртуалізіруя зовнішні джерела даних без необхідності їх прямого імпорту в реляційну базу, і потім дозволяє звертатися до цих даних із запитами.

Таким чином, дані накопичуються в своєму природному форматі, не обов'язково реляційні бази, але і ті, що можуть бути представлені у вигляді віртуальних таблиць. Віртуалізація дозволяє інтегрувати дані різного формату,

з різнорідних джерел і місць зберігання без їх реплікації і переміщення, створюючи єдину віртуальну матрицю даних.

Для подальшої розробки системи було повністю спроектовано базу даних. Кінцева версія БД має такий вигляд (рисунок 3.2):

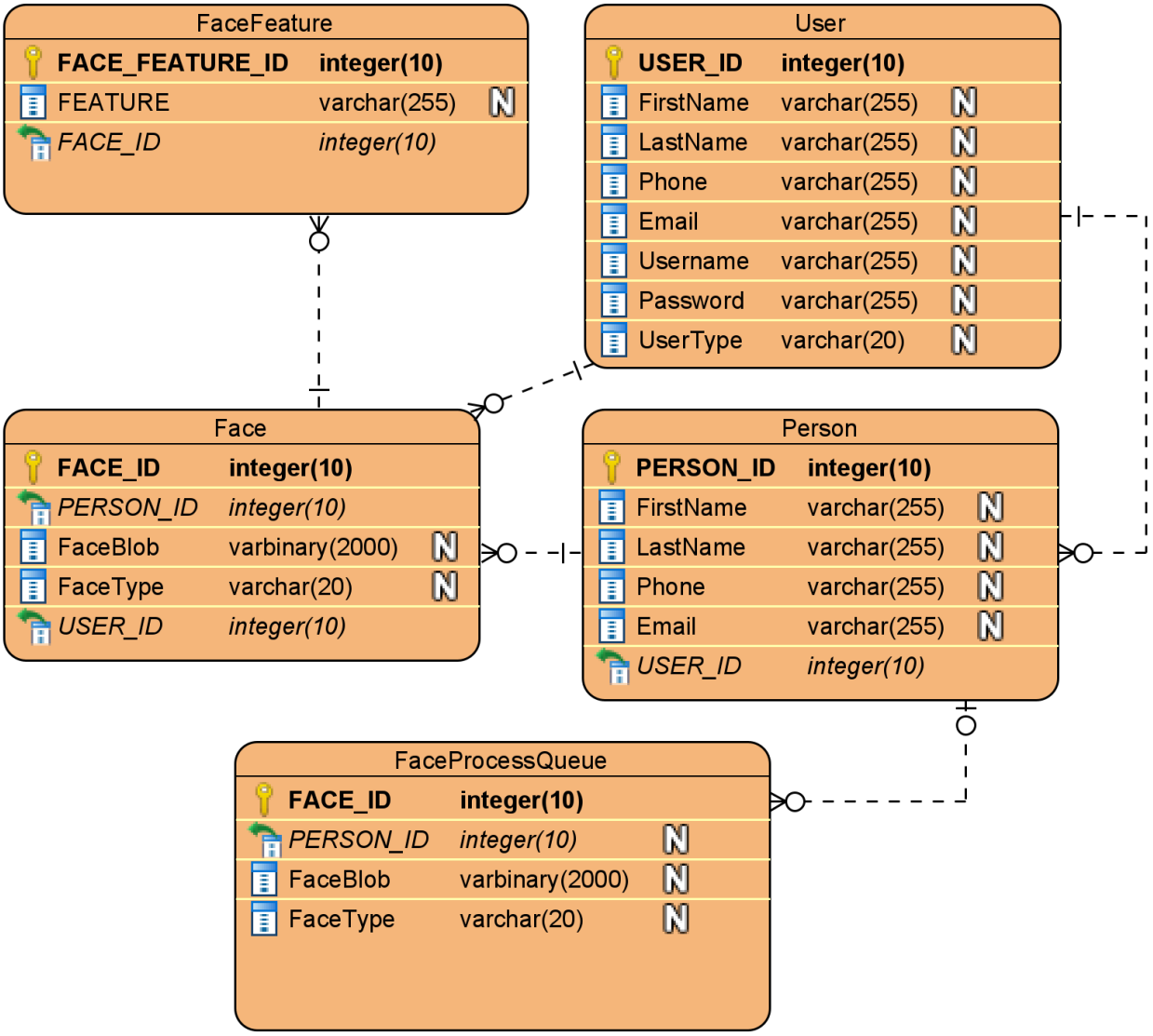


Рисунок 3.2– Діаграма бази даних

Розглянемо схему детальніше та опишемо кожне поле в усіх таблицях.

1) "User" – таблиця користувачів (містить Unique constraint для перевірки значення поля “Email” та "Username" на унікальність):

"USER_ID" унікальний ідентифікатор користувача,

"FirstName, LastName" – поля для імені,

"UserType" – ти коритсувача (адміністратор, чи звичайний користувач)

"Password", - пароль користувача

"Phone","Email" – контактні данні

2) "Person" – данні людини яка зберігається в базі

"PERSON_ID" унікальний ідентифікатор особи,

"FirstName, LastName" – поля для імені,

"Phone","Email" – контактні дані особи

"USER_ID" унікальний ідентифікатор користувача, який додав цю особу до бази. Може бути тільки 1 користувач що додав.

3) "Face" - таблиця облич користувачів. Один користувач може мати декілька його обличь в базі. Тим більше, це необхідно, аби розпізнавати його обличчя з максимальною точністю Тут можуть міститись данні як про ціле обличчя так і його частину.

"FACE_ID" унікальний ідентифікатор обличчя,

"FaceBlob" – зображення обличчя зжате до 256кб,

"FaceType" -тип зображення (все обличчя чи верхня частина),

"USER_ID" зовнішній ключ. Ідентифікатор користувача, який додав це обличчя до бази. Може бути тільки 1 користувач що додав.

4) " FaceFeature" - таблиця рис облич. з того самого 128 вимірного вектору

"FACE_FEATURE_ID" унікальний ідентифікатор риси,

"Feature" – значення риси,

"FACE_ID" зовнішній ключ. Ідентифікатор обличчя до якого належить риса. Риса може належати тільки до 1 обличчя.

4) " FaceProcessQueue" - таблиця яка служить до тимчасового зберігання зображень облич, з яких ще не отримано вектор рис. Завдяки цый таблиці відбувається взаємодія між сервером API ASP.NET Core та скриптом Python для розпізнавання облич.

"FACE_ID" унікальний ідентифікатор обличчя,

"FaceBlob" – зображення обличчя,

"FaceType" -тип зображення (все обличчя чи верхня частина).

Лістинг скрипту для створення бази даних подано у додатку В.

3.3 Розробка клієнтського Web додатку

Для розробки було обрано відомий фреймворк Angular.

Це фреймворк з відкритим кодом, який дозволяє створювати адаптивні та структуровані веб додатки. Після запуску в 2009 році Angular.js швидко набрав популярність і не збираєтьсяздавати позиції - він залишається досить затребуваним JavaScript-фреймворком на GitHub (у тому числі як за кількістю коду, так і за кількістю «зірок»).

Розглянемо більш детально, чому було обрано саме AngularJS і які переваги має даний фреймворк.Першою перевагою є використання декларативного стилю коду.

При створенні шаблонів в Angular.js застосовується декларативна парадигма програмування. Це робить код більш легковажним, полегшує його читання і підтримку, так як описується необхідний кінцевий результат, а не всі кроки по його досягненню. Наприклад, на рисунках 3.3 та 3.4 показано порівняння коду написаного на чистому JS та з використанням фреймворку Angular.

```
let facesList = document.getElementById( elementId: 'faces-list');
faces.forEach(function(e){
    let faceEntry = document.createElement( tagName: 'li'),
        faceName = document.createElement( tagName: 'span'),
        faceScore = document.createElement( tagName: 'span');
    faceName.innerHTML = e.name;
    faceScore.innerHTML = e.score;
    faceEntry.appendChild(faceName);
    faceEntry.appendChild(faceScore);
    faceList.appendChild(faceEntry);
});
```

Рисунок3.3– Фрагмент коду мовою JS

```
<div ng-controller="facesController as facesCrtl">
    <div ng-repeat="face in facesCrtl.faces">
        Name: <span class="name">{{face.name}}</span>
        Title: <span class="title">{{face.title}}</span>
    </div>
</div>
```

Рисунок3.4– Фрагмент коду в фреймворку Angular

HTML використовується в якості мови шаблонів в Angular.js. Він розширюється за допомогою директив, які додають в код відомості про необхідний поведінці (наприклад, про необхідність завантажити певний модуль відразу після завантаження сторінки). Директиви дозволяють вам сконцентруватися на опрацюванні логіки і працювати більш продуктивно. Їх можна використовувати повторно, що також підвищує читабельність коду.

У AngularJS, так само як і в серверній частині використовується схема MVC, що розділяє логіку, графічну реалізацію і дані програми (рисунок 3.5).

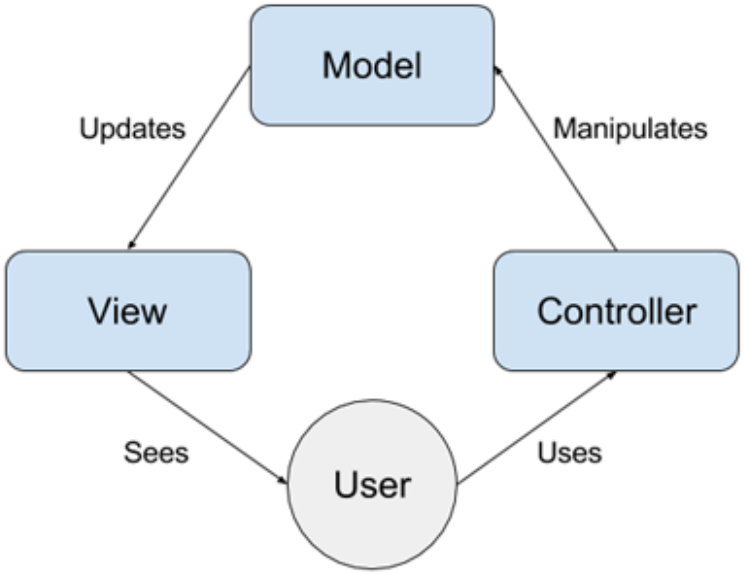


Рисунок 3.5 – Архітектура клієнтського веб-додатку

Лістинг фрагментів коду для розпізнання обличчя на зображенні подано у додатку Г, а фрагменти коду для розпізнавання з відеопотоку – у додатку Д.

3.4 Розробка клієнтського десктоп додатку

Клієнтський десктоп-додаток побудований з використанням фреймворку PyQT5, який є оболонкоюдля бібліотеки QT на мові програмування Python.PyQt5 є одним з найбільш часто використовуваних модулів для створення GUI додатків в Python

Фреймворк PyQt поєднує в собі всі переваги Qt і Python. PyQt являє собою набір прив'язок Python v2 і v3 для інфраструктури додатків Qt і працює на всіх платформах, підтримуваних Qt, включаючи Windows, OS X, Linux, iOS і Android. Прив'язки реалізовані у вигляді набору модулів Python і містять більше 1000 класів.

PyQt має подвійну ліцензію на всіх підтримуваних платформах в рамках GNU GPL v3 і комерційної ліцензії Riverbank. На відміну від Qt, PyQt недоступний в LGPL. Встановлення PyQt5 відбувається через pip. Щоб встановити PyQt5 за допомогою pip, потрібно виконати наступну команду:sudo pip3 install PyQt5.

Фреймворк досить еластичний і дозволяє швидко розбудовувати архітектуру десктопних додатків. Загальну схему архітектури десктопного додатку подано на рисунку 3.6.

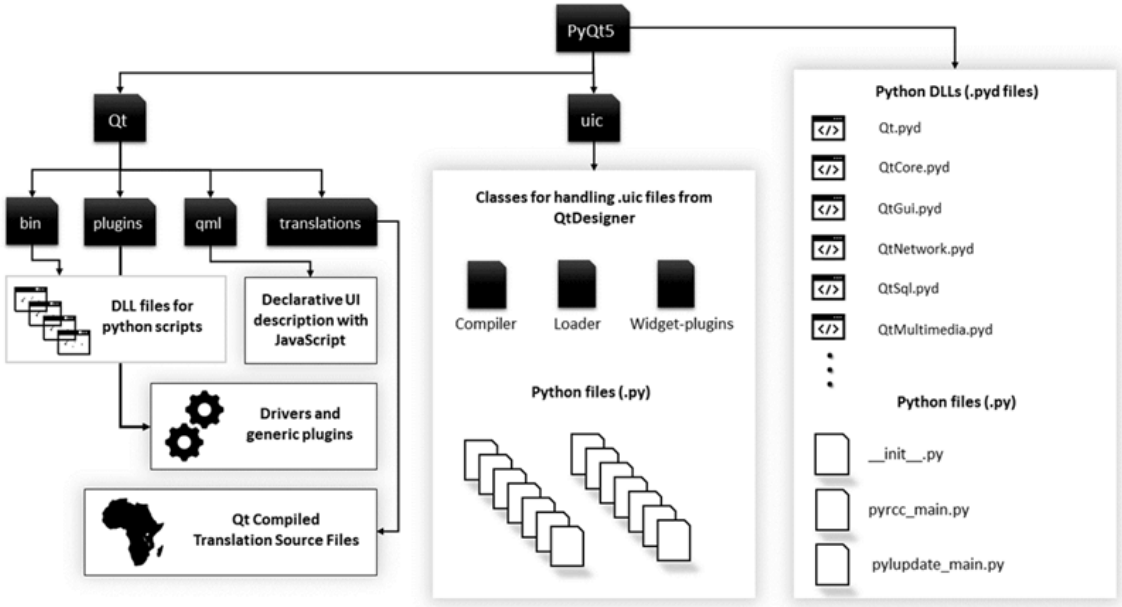


Рисунок 3.6– Архітектура клієнтського веб-додатку

Основні модулі, що використовуються Qt Python, зокрема PyQt5 є:

Основний Qt модуль: він поєднує всі класи / модулі, згадані нижче, в один модуль. Це значно збільшує обсяг пам'яті, що використовується додатком. Однак простіше керувати фреймворком, імпортуючи лише один модуль.

QtCore містить основні неграфічні класи, що використовуються іншими модулями. Тут реалізовано цикл подій Qt, сигнали та підключення до слотів тощо.

QtWidgets містить більшість віджетів, доступних у Pyqt5.QtGui містить компоненти графічного інтерфейсу та розширює модуль QtCore.QtNetwork містить класи, що використовуються для реалізації мережевого програмування через Qt. Він підтримує сервери TCP, сокети TCP, сокети UDP, обробку SSL, мережеві сесії та пошук DNS.QtMultimedia забезпечує мультимедійну функціональність низького рівня.QtSql реалізує інтеграцію баз даних для баз даних SQL. Підтримує ODBC, MySQL, Oracle, SQLite та PostgreSQL.

3.5 Висновки

У третьому розділі магістерської кваліфікаційної роботи було проаналізовано і вибрано технології реалізації кожного з компонентів системи.

Оскільки для додатку було обрано мікросервісну архітектуру, то бекенд та фронтенд розроблялися окремо.

Додатково розроблено десктопний застосунок.

У розробці клієнтської частини для веб версії використано фреймворк Angular, а для десктопної версії PyQT5 Python разом з QT builder та PyQT tools.

Для розробки серверної частини обрано ASP.NET core – популярний фреймворк для розробки на C#, який надає широкий спектр можливостей у написанні програмного коду.

Також було розроблено структуру бази даних веб-додатку. База даних обслуговується за допомогою SQLServer.

Було визначено основні функціональна та нефункціональні вимоги веб-додатку, а також представлено діаграму процесу розпізнавання обличчя.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів та засобів тестування

Залежно від доступу розробника тестів до вихідного коду програми, що тестується розрізняють «тестування білого ящика» і «тестування чорного ящика».

При тестуванні білого ящика (також кажуть - прозорого ящика), розробник тесту має доступ до вихідного коду програм і може писати код, який пов'язаний з бібліотеками тестованого програмного забезпечення. Це типово для модульного тестування, при якому тестуються лише окремі частини системи. Воно забезпечує те, що компоненти конструкції - працездатні і стійкі, до певної міри. При тестуванні білого ящика використовуються метрики покриття коду. При тестуванні чорного ящика, тестувальник має доступ до програми тільки через ті ж інтерфейси, що і замовник або користувач, або через зовнішні інтерфейси, що дозволяють іншого комп'ютера або іншому процесу підключитися до системи для тестування. Наприклад, тестувальник, тестуючи модуль може віртуально натискати клавіші або кнопки миші в програмі за допомогою механізму взаємодії процесів, з упевненістю в тому, чи все йде правильно, що ці події викликають той же відгук, що й реальні натискання клавіш і кнопок миші.

Як правило, тестування чорного ящика ведеться з використанням специфікацій або інших документів, що описують вимоги до системи. Зазвичай в даному виді тестування критерій покриття складається з покриття структури вхідних даних, покриття вимог і покриття моделі (в тестуванні на основі моделей).При тестуванні сірого ящика розробник тесту має доступ до вихідного

коду, але при безпосередньому виконанні тестів доступ до коду, як правило, не потрібно.

Якщо «альфа-» і «бета-тестування» відносяться до стадій до випуску продукту, то тестування «білого ящика» і «чорного ящика» має відношення до способів, якими тестувальник досягає мети.Бета-тестування в цілому обмежена технікою чорного ящика (хоча постійна частина тестувальників зазвичай продовжує тестування білого ящика паралельно бета-тестування). Таким чином, термін «бета-тестування» може вказувати на стан програми (ближче до випуску ніж «альфа»), або може вказувати на деяку групу тестувальників і процес, що виконується цією групою. Тобто, тестувальник може продовжувати роботу з тестування білого ящика, хоча програма вже «бета-стадії», але в цьому випадку він не є частиною «бета-тестування».

Тестування чорного ящика або поведінковий тестування - це така стратегія тестування функціонального поведінки об'єкта (програми, системи) з точки зору зовнішнього світу, при якому не використовується знання про внутрішній устрій тестованого об'єкта. Під стратегією розуміються систематичні методи відбору і створення тестів для тестового набору. Стратегія поведінкового тесту виходить з технічних вимог і їх специфікацій.

Структурне тестування – тестування коду на предмет логіки роботи програми і коректності її роботи з точки зору компілятора того мови, на якому вона писалась. Тестування за стратегією білого ящика, також зване технікою тестування, керованої логікою програми, дозволяє перевіритивнутрішню структуру програми. Виходячи з цієї стратегії тестувальник отримує тестові дані шляхом аналізу логіки роботи програми.

Техніка білого ящика включає в себе наступні методи тестування:

- покриття рішень;
- покриття умов;

- покриття рішень і умов;
- комбінаційне покриття умов.

Таким чином, для тестування системиу магістерській кваліфікаційній роботі було обрано динамічну техніку тестування, яка базується на специфікації, тобто так зване тестування «чорної скриньки»

4.2 Тестування програмного забезпечення

Для початку було вирішено протестувати просту детекцію обличчя. Для цього потрібно перейти до вкладки детекція обличчя. На рисунку 4.1 можна побачити детекцію обличь з відеопотоку.

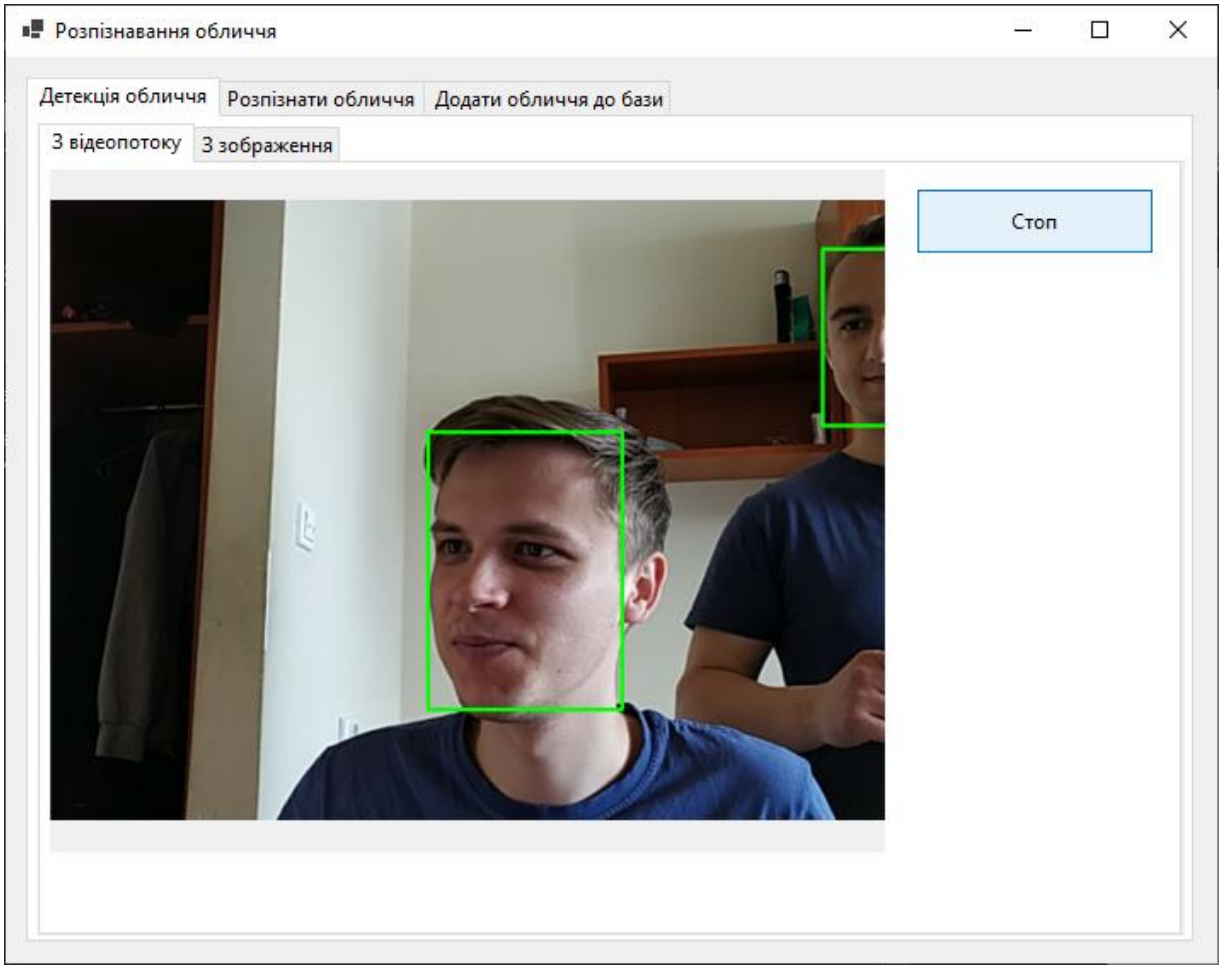


Рисунок 4.1 – Детекція облич з відеопотоку

Також можна робити детекцію з окремого зображення. Пункт меню “З зображення” забезпечує розпізнавання з окремого зображення обличчя людини(рисунок 4.2).

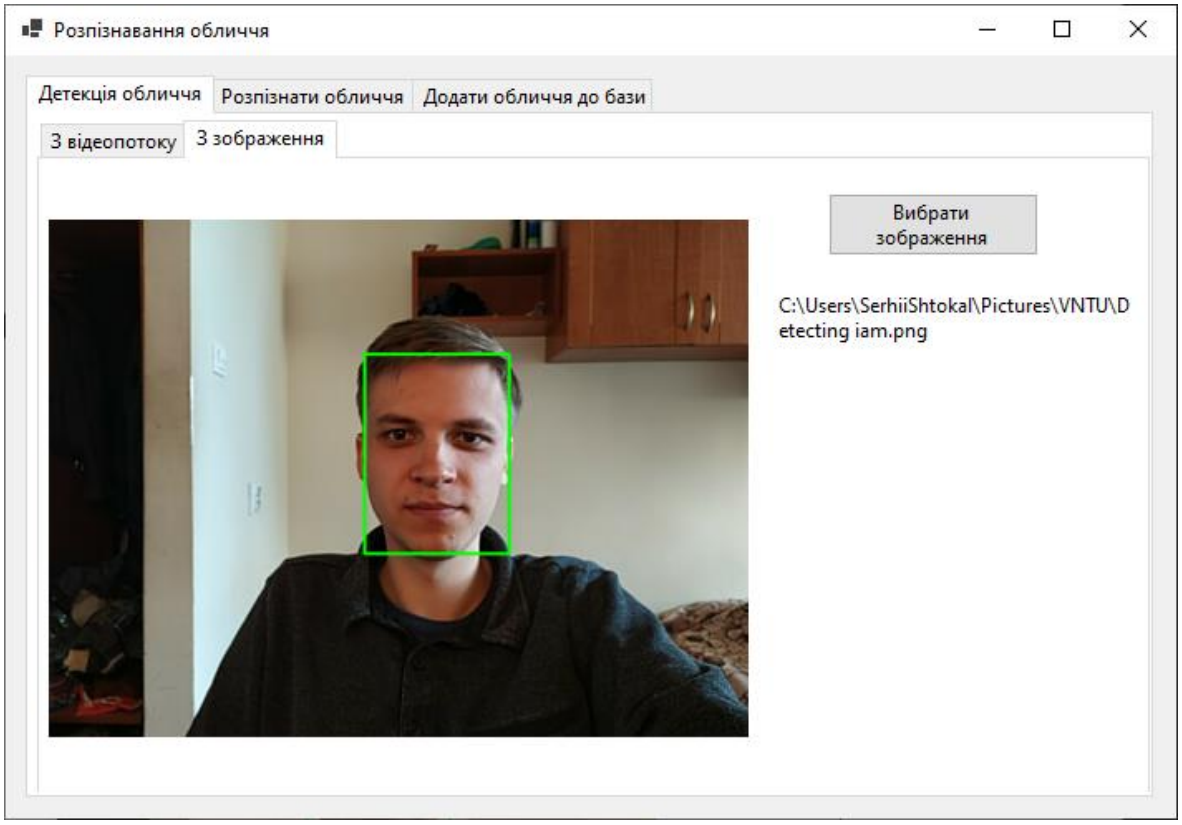


Рисунок 4.2– Детекція обличчя з зображення

Далі було протестовано функціонал розпізнавання обличчя. Розпізнавання облич з відеопотоку показано на рисунку 4.3.

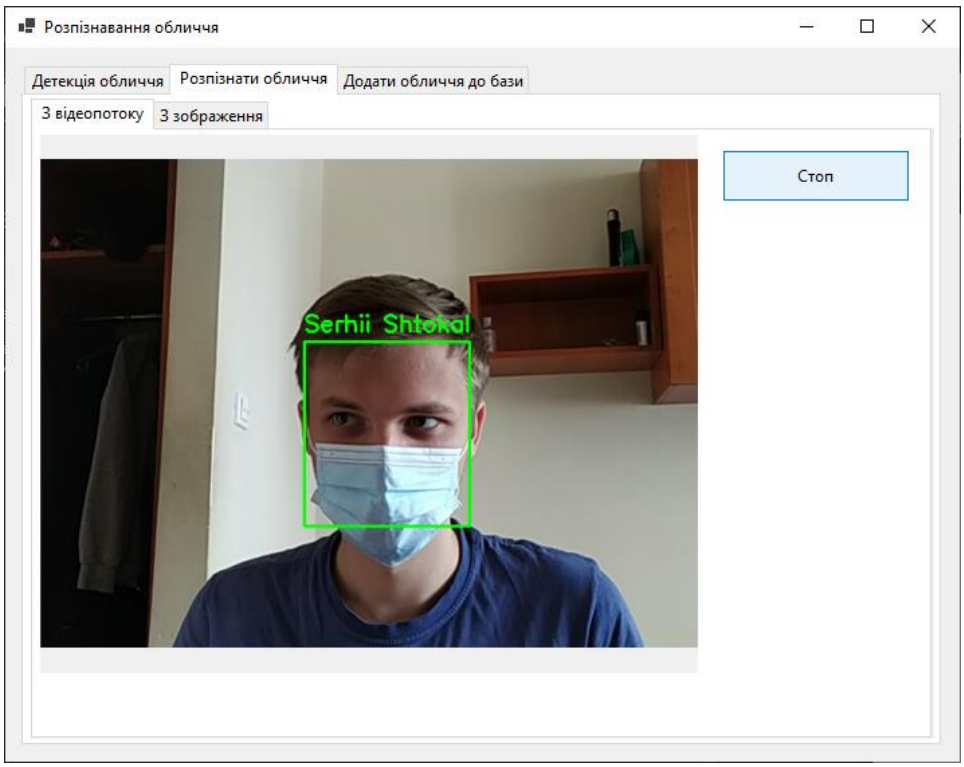


Рисунок 4.3 – Розпізнання обличчя з відеопотоку

Додатково реалізовано розпізнавання обличчя з зображення. Система підтримує різні формати зображень, зокрема .png, .jpg та інші.

На рисунку 4.4 вибрано зображення дещо відредагованого лиця за допомогою програми FaceApp, незважаючи на змінення лиця, система розпізнала обличчяправильно.

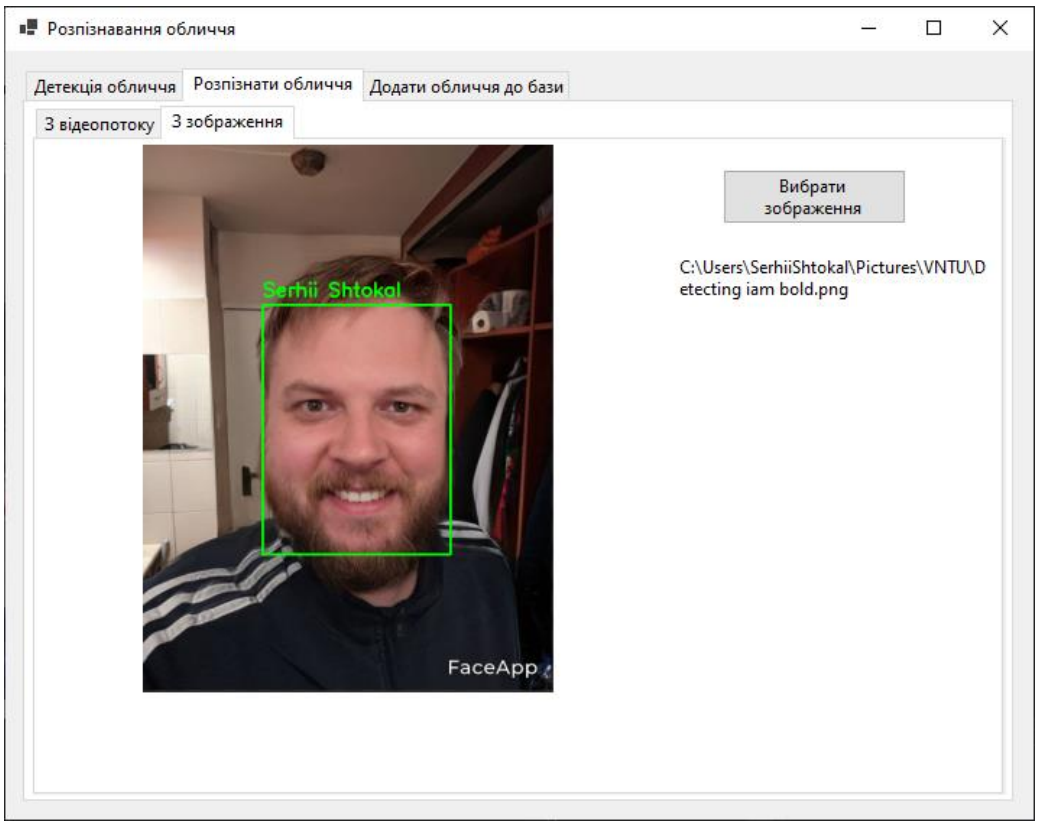


Рисунок 4.4 – Розпізнавання обличчя з зображення

Пункт меню “Додати обличчя до бази” відповідає за додавання нових облич до бази. Так само як детекція і розпізнавання, додавання може здійснюватися у двох режимах: з відеопотоку або з зображення.

Додавання облич з відеопотоку показано на рисунку 4.5.

У випадку коли система не в змозі розпізнати обличчя – воно сильно перекрито чи положення обличчя нестандартне, то система не зможе його розпізнати, у зв'язку з малою кількістю набору зображень обличчя для даної особи. Приклад роботи програми, коли обличчя не розпізнане, наведено на рисунку 4.6.

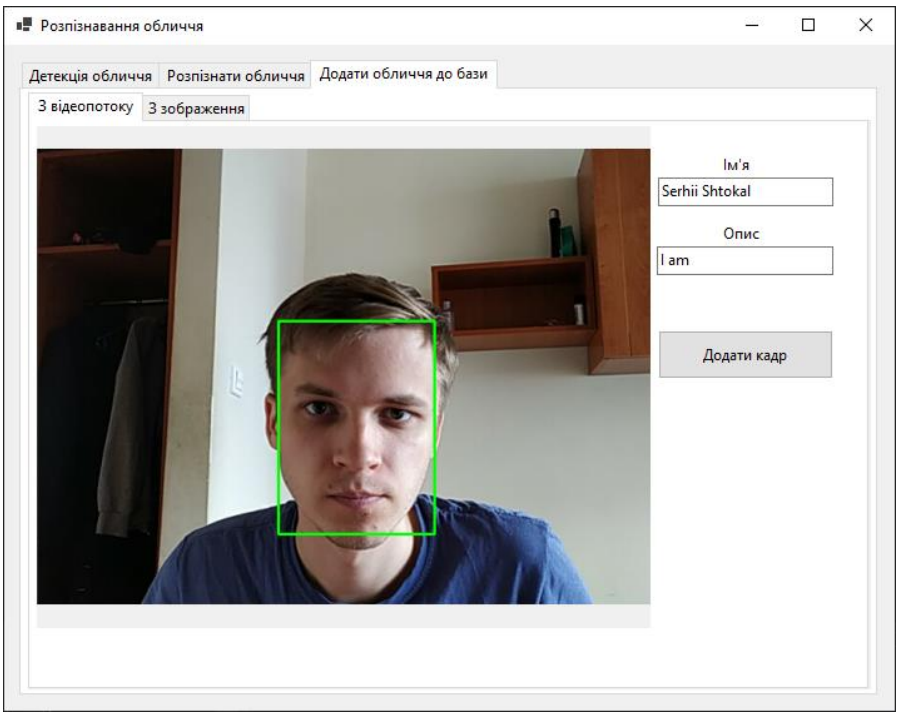


Рисунок 4.5–Додавання обличчя з відеопотоку

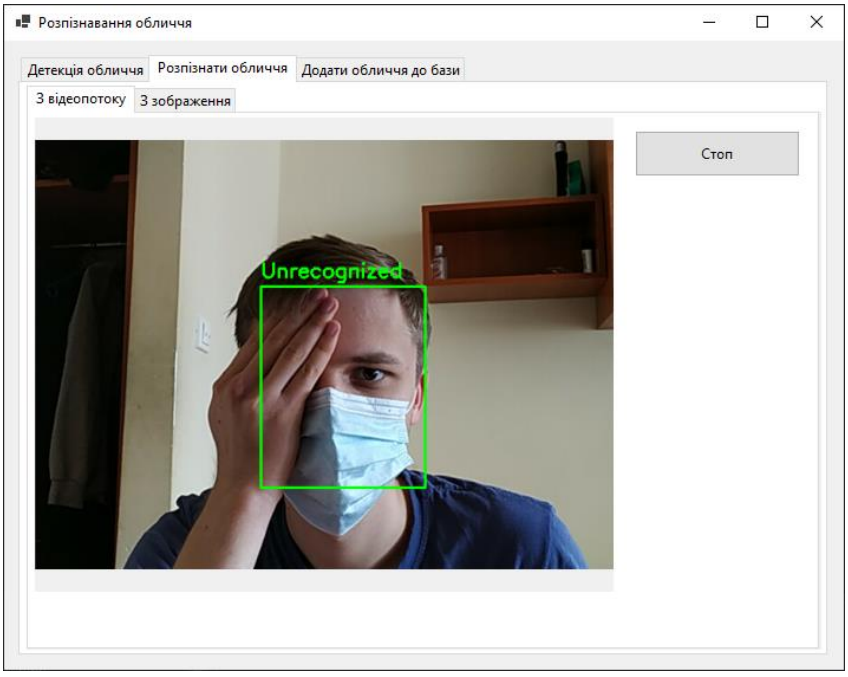


Рисунок 4.6–Приклад роботи програми, коли обличчя не розпізнане

Використання зображень може спростити додавання нових облич до бази, адже не вимагається особиста присутність людини – достатньо лише мати її фотографію. Таким чином, при першій спробі детекції на відео таке обличчя буде одразу розпізнано.

4.3 Розробка інструкції користувача

Інструкція користувача – це документ, призначення якого надати людям допомогу в використанні системою. Документ входить до складу технічної документації на систему і, як правило, готується технічним працівником або програмістом. Інструкція дозволяє допомогти вирішити певну проблему, або зменшити час ознайомлення користувачів з програмним продуктом, за рахунок

Користуватися системою можуть лише зареєстровані користувачі.

Для того щоб створити новий обліковий запис необхідно на головному екрані натиснути кнопку “Реєстрація” і заповнити форму з даними користувача.Щоб увійти до системи необхідно натиснути кнопку “Вхід”.

Користувач, що увійшов до системи може налаштувати головний екран. Існує три режими:

- детекція обличчя;
- розпізнавання обличчя;
- додавання нового обличчя до бази.

На рисунку 4.7 показано діаграму Use-case, що описує роботу з цими трьома режимами.

На екрані детекції потрібно натиснути кнопку “Старт”, щоб почати детекцію, а також “Стоп”, щоб зупинити. Режим детекції працює таким чином, що дозволяє захоплювати будь-яке обличчя, що з'явиться в полі зору камери (під час детекції з відеопотоку) або ж в межах одного зображення (під час детекції з зображення). Щоб завантажити зображення необхідно натиснути кнопку “Вибрати зображення” і у новому вікні вибрати зображення з каталогу або зробити фото одразу, використовуючи камеру.

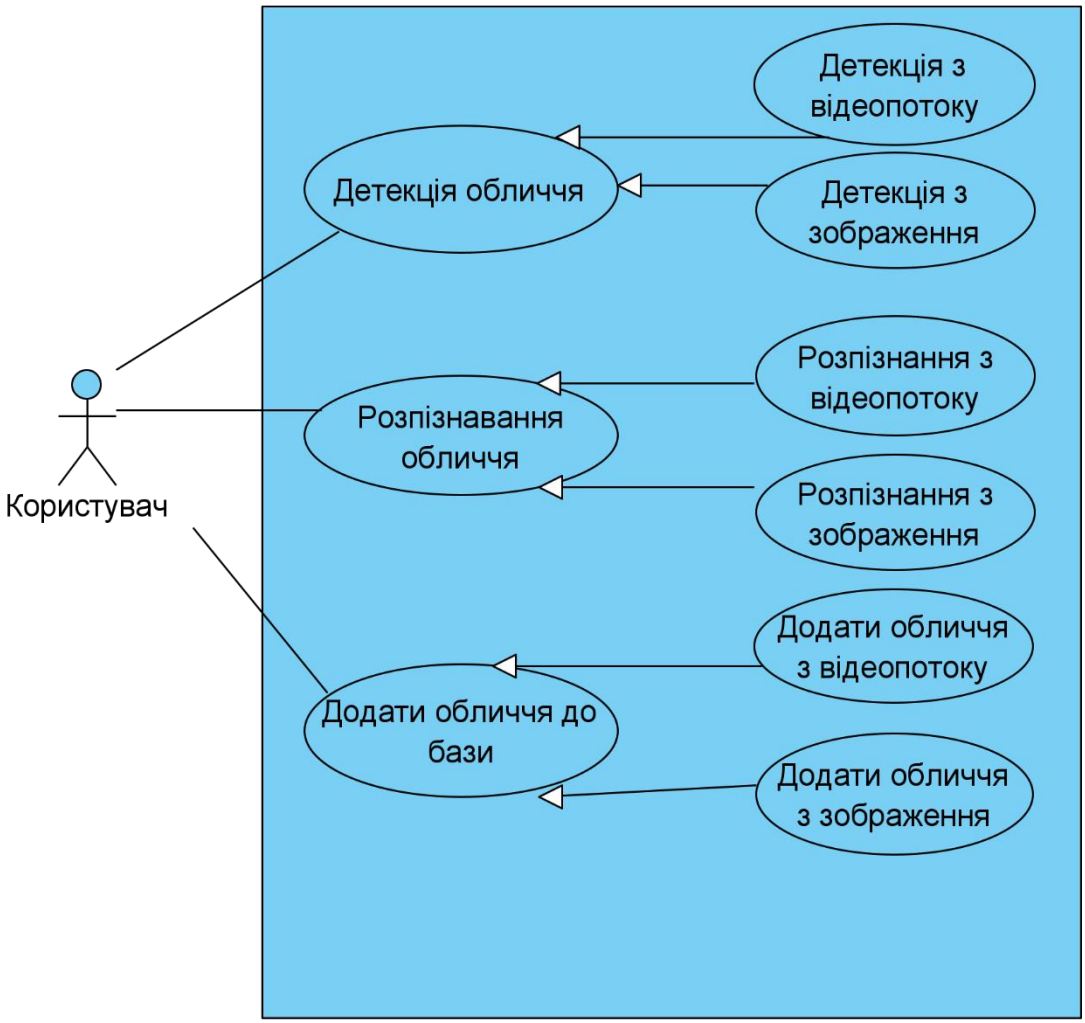


Рисунок 4.7 –Діаграма use-case

Екран розпізнавання виглядає подібним чином, розпізнавання можна почати/зупинити натисненням кнопок “Стоп” та “Старт” відповідно. Так само існує можливість завантажити зображення до розпізнавання обличчя.Під час розпізнавання система шукає захоплене на відео чи зображенню обличчя в базі і якщо його знаходить, то на обличчі з'являється зелена рамка з вказанням особистих даних людини, а саме - ім'я та прізвище. Процес супроводжується звуковим сигналом. Якщо обличчя не існує в базі то лунає відповідний звуковий сигнал, о на обличчі з'являється рамка з надписом “не знайдено”.

На екрані додавання обличчя до бази реалізована можливість завантаження нових даних користувачів. Як в попередніх двох режимах існує можливість додати обличчя з відеопотоку, так і з зображення. Система

розпізнає, а потім запам'ятовує нове обличчя. Необхідно також подати особисті дані людини. Такі поля як ім'я та прізвища є обов'язковими.

Щоб зупинити роботу з програмою достатньо натиснути кнопку “Стоп” у будь-якому з режимів.

Таким чином, всі необхідні інструкції будуть надані користувачеві перед початком роботи з системою.

4.4 Висновки

У четвертому розділі магістерської кваліфікаційної роботи було проаналізовано різні методи тестування, обрано найбільш вдалий метод тестування – метод чорної скриньки.Система для кожного з розглянутих тестових сценаріїв працює правильно. Було проведено опис роботи головних сценаріїв веб-додатку. Розроблено інструкцію користувача.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

З метою проведення технологічного аудиту було прийнято рішення провести оцінювання комерційного потенціалу розробки. З цією метою було залучено 2-х незалежних експертів. Такими експертами будуть Войтко Вікторія Володимирівна (к.т.н., доц. кафедри ПЗ ВНТУ), а також Хошаба Олександр Мирославович (к.т.н., доц. кафедри ПЗ ВНТУ).

Оцінювання комерційного потенціалу розробки здійснюється за 12-ма критеріями за 5-ти бальною шкалою.

Критерії проведення оцінювання комерційного потенціалу наведено в таблиці 5.1.

Таблиця 5.1 – Критерії оцінювання комерційного потенціалу розробки бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-тер.	0	1	2	3	4
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна Конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне значне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці 5.1

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні Незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.2.

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	Войтко В. В.	Хошаба О.М.
	Бали, виставлені експертами:	
1	3	4
2	3	2
3	2	3
4	4	4
5	2	3
6	3	4
7	4	3
8	4	2
9	3	3
10	2	3
11	4	4
12	2	2
Сума балів	$СБ_1 = 36$	$СБ_2 = 37$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 36{,}5$	

Отже, з отриманих даних таблиці 5.2 видно, що нова розробка має достатній рівень комерційного потенціалу.

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи

Проаналізовано витрати необхідні для розробки програмного продукту.

Основна заробітна плата для розробників визначається за формулою (5.1):

$$З_о = \qquad (5.1)$$

де М- місячний посадовий оклад конкретного розробника;

- кількість робочих днів у місяці, = 22 дні;

t - число днів роботи розробника, t = 45 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці5.3.

Таблиця 5.3 – Розрахунки основної заробітної плати

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	30000	1363,63	5	6818,15
Full Stack програміст	25000	1136,36	20	22727,2
Всього:				29545,35

Розрахуємо додаткову заробітну плату:

$$Здод = 0{,}1 \cdot 29545{,}35 = 2954{,}535 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 22% від суми їхньої основної та додаткової заробітної плати (вираз 5.2):

$$Нзп = (Зо + Зр) \cdot \frac{\beta}{100}, \qquad (5.2)$$

$$Нзп = (29545{,}35 + 2954{,}535) \cdot \frac{22}{100} = 7149{,}97 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою 5.3, а результати подані у таблиці 5.4:

$$A = \frac{Ц \cdot}{1} \quad (5.3)$$

де Ц – балансова вартість обладнання, грн;

– річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 5.4 – Розрахунок амортизаційних відрахувань

Найменування	Ціна, грн.	Норма амортизації, %	Термін використання, м.	Сума амортизації
ПК + прилади керування	10000	20	3	500
Прилади маніпуляції ПК	1000	20	3	50
Всього	550			

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою 5.4, а результати запишемо до таблиці 5.5:

$$K = \sum_1^n H_i \cdot] \quad (5.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (нехай Кі = 1,1).

Таблиця 5.5 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	200	1	200
Фліпчарт	шт.	350	1	350
Маркери	шт.	25	4	100
Всього з урахуванням транспортних витрат				750

Витрати на силову електроенергію розраховуються за формулою 5.5:

$$В_е = В \cdot П \cdot \ ; \qquad (5.5)$$

де В – вартість 1кВт-години електроенергії (В=9,5 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=250 год.);

– коефіцієнт використання потужності (< 1, $К_п$).

$$Ве = 9{,}5 \cdot 0{,}6 \cdot 250 \cdot 0{,}8 = 1140 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто (формула 5.6):

$$В_{ін} = (1..3) \cdot (З_о - \qquad (5.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (29545{,}35 + 2954{,}535) = 32\,49 \text{ (грн).}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 295+ 295+ 71+550+750+1140+ 32 49 = 74 589,74 (грн.)

Розрахуємо загальну вартість наукової роботи $В_{заг}$ за формулою 5.7:

$$В_{заг} \qquad (5.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{74\,589{,}74}{1} = 74\,589{,}74$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою 5.8:

$$ЗВ \tag{5.8}$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{74\,589{,}74}{0{,}9} = 82\,8 \text{(грн.)}$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою 5.9:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П \tag{5.9}$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення програмного продукту зменшаться на 60 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 60 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 450 користувачів, протягом другого року – на 400 користувачів, протягом третього року – 350 користувачів. Реалізація програмного продукту до впровадження результатів наукової розробки складала 50 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 100 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_{2020} = 60 \cdot 50 + (100 + 60) \cdot 450 = 75000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_{2021} = 60 \cdot 50 + (100 + 60) \cdot (450 + 400) = 139000 \text{ грн.}$$

Протягом третього року:

$$\Delta П_{2022} = 60 \cdot 50 + (100 + 60) \cdot (450 + 400 + 350) = 195000 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність вкладених інвестицій розраховується за формулою 5.10:

$$E_{абс} = (ПП - \quad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Розрахуємо вартість чистих прибутків за формулою 5.11:

$$ПП = \sum_1^m \quad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{82\,877{,}48}{(1+0{,}1)^0} + \frac{75000}{(1+0{,}1)^2} + \frac{139000}{(1+0{,}1)^3} + \frac{195000}{(1+0{,}1)^4} = 3824 \text{(грн.)}$$

Тоді розрахуємо

$$E_{абс} = 382481{,}32 - 82\,877{,}48 = 299\,6 \text{грн.}$$

Оскільки $E_{аі}$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою 5.12:

$$E_в = \sqrt[T]{1 + \frac{E_{}}{P}} \qquad (5.12)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{299\,603{,}84}{82\,877{,}48}} - 1 = \text{або } 58\ \%$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою 5.13:

OBJ,(5.13)

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в = 58\% > 30\%$, то інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проєкту інвестицій. Термін окупності вкладених у реалізацію наукового проєкту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{0{,}44} = 2{,}27 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

5.5 Висновки

1. Оцінювання комерційного потенціалу розробки показало, що розробка має високий рівень комерційного потенціалу.
2. Розраховано витрати на виконання науково-дослідної та конструкторської-технологічної роботи
3. Розраховано збільшення чистого прибутку протягом трьох років.
4. Обрахунок терміну окупності розробки показав, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

У результаті виконання магістерської кваліфікаційноїроботи було розроблено функціонал системи для розпізнавання обличчя людини.

Детальний аналіз предметної області показав, що проблема потребує вирішення. Було виконано аналіз відомих аналогів даного проекту, після чого створено перелік завдань та план дій для розробки програмного продукту.

Розроблено загальну функціональну структурну модель, яка описує основні процеси системи.

Під час розробки було реалізовано:

- розроблено метод виявлення та обробки рис верхньої частини обличчя
- розроблено метод розпізнавання обличчя, що поєднує функціонал методів розпізнавання, рис верхньої частини обличчя та рис всього обличчя
- модель системи розпізнавання обличчя людини;
- дизайн інтерфейсу для клієнтських додатків;
- програмну реалізацію системи розпізнавання обличчя людини;
- архітектуру бази даних веб-системи;
- розроблено серверну частини та програмні засоби системи, що містить API для роботи з клієнтськими додатками для веб та десктоп версії;

Магістерську кваліфікаційнуроботу було оформлено відповідно до вимог [30].Розроблена система була протестована за допомогою технології «чорної скриньки». Тестування показало коректність відображення інформації та довело працездатність системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Распознавание лиц в период пандемии. Как справляются с «хакерской атакой» системы видеоаналитики. URL: https://kod.ru/raspoznavanie-lits-v-pieriod-pandemii/.
2. Штокал С.С., Войтко В.В., Хошаба О.М. Тези доповідей XII науково-практичної конференції"Інформаційно-комп'ютерні технології – 2021 (ІКТ-2021)". м. Житомир, 01-03 квітня 2021 р., ст. 32-33.
3. M. Haischer et al., Who is wearing a mask? Gender-, age-, and location-related differences during the COVID-19 pandemic, PLOS ONE, 2020 p., ст. 341-347.
4. Use Face ID on your iPhone or iPad Pro, Apple Support. URL: https://support.apple.com/en-us/HT208109
5. Home – Face2Gene. URL: https://www.face2gene.com.
6. Gurovich, Y. et al. DeepGestalt—identifying rare genetic syndromes using deep learning. URL: https://arxiv.org/abs/1801.07637.
7. Facial recognition: School ID checks lead to GDPR fine. URL: https://www.bbc.com/news/technology-49489154.
8. New York Creates First-in-the-Nation Moratorium on Facial Recognition in Schools, New York Civil Liberties Union, URL: https://www.nyclu.org/en/press-releases/new-york-creates-first-nation-moratorium-facial-recognition-schools.
9. Delta expands optional facial recognition boarding to new airports, more customers. URL: https://news.delta.com/delta-expands-optional-facial-recognition-boarding-new-airports-more-customers.
10. FaceApp - AI Face Editor. URL: Available: https://www.faceapp.com.

11. Q. Jin, J. Zhao and Y. Zhang, Facial feature extraction with a depth AAM algorithm, 2012 9th International Conference on Fuzzy Systems and Knowledge Discovery, 2012.

12. J. Grewal, M. Krzywinski and N. Altman, Markov models — hidden Markov models, 2019, ст. 795-796.

13. P. Sang, L. Wang and J. Cao, Parametric functional principal component analysis, 2017, ст. 802-810.

14. W. Yang and H. Wu, "Regularized complete linear discriminant analysis", Neurocomputing, vol. 137, pp. 185-191, 2014. Available: 10.1016/j.neucom.2013.08.048.

15. Animetrics Face Recognition API. URL: http://animetrics.com/?content=products/aPIs.

16. Luxand.cloud - face recognition API, Luxand.cloud. URL: https://luxand.cloud/.

17. AI Solutions Provider | SenseTime, Sensetime.com. 2021. URL: https://www.sensetime.com/. .

18. B. Harwani, Introduction to Python programming and developing GUI applications with PyQT. Boston, MA: Course Technology, 2012.

19. B. Cherny, Programming TypeScript: Making Your JavaScript Applications Scale, O'Reilly Media.

20. Angular contributors, Angular.io, 2021. URL: https://angular.io/about?group=Angular. .

21. D. Strauss, Creating ASP. NET core web applications. Berkeley, CA: Apress, 2021.

22. H. Schwichtenberg, Modern Data Access with Entity Framework Core: Database Programming Techniques for .NET, .NET Core, UWP, and Xamarin with C#, 1st ed. Apress, 2017.

23. Azure.microsoft.com. 2021. Cloud Computing Services | Microsoft Azure. URL: https://azure.microsoft.com.

24. M. Owens and G. Allen, The Definitive Guide to SQLite. New York: Apress L.P, 2010.

25. P. DuBois, MySQL. Indianapolis, Addison-Wesley, 2013.

26. G. Smith, PostgreSQL 9.0 high performance. Birmingham

27. J. Guay Paz, Microsoft Azure Cosmos DB Revealed.

28. F. Schroff, D. Kalenichenko, J. Philbin, FaceNet: A unified embedding for face recognition and clustering, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, ст. 815-823

29. K. Sanal Kumar and R. Bhavani, Activity Recognition in Egocentric video using SVM, kNN and Combined SVMkNN Classifiers, ст. 225. URL: 10.1088/1757-899x/225/1/012226.

30. О. Н. Романюк, Р. Р. Обертюх, Т. О. Савчук, Л. П. Громова. Положення про кваліфікаційну роботу у Вінницькому національному технічному університеті. Вінниця, 2015. 27 с.

ДОДАТКИ

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
17 лютого 2021 року

Технічне завдання
на магістерську кваліфікаційну роботу «Розробка методів та програмного додатку розпізнавання обличчя людини»
за спеціальністю121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
к.т.н., доцент Хошаба О.М.
18 лютого 2021 року
Виконав:
студент гр. 1ПІ-19м Штокал С.С.
18 лютого 2021 року

Вінниця – 2021 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів та програмного додатку розпізнавання обличчя людини».

Галузь застосування – розпізнавання образів.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР тта наказ від 9 березня 2021 року № 65 ректора ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності розпізнавання обличчя людини в масці за рахунок методів розпізнавання рис верхньої частини обличчя, що забезпечує точність розпізнавання в умовах неповної видимості обличчя.

Призначення роботи – розробка методів і програмних засобів для розпізнавання обличчя людини.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

4. Технічні вимоги

Оскільки проект ділиться на 2 частини (клієнт, сервер) розглянемо вимоги до кожного.

1. Клієнт. Клієнтська частина складається з двох додатків: веб-версія та десктопна аплікація. Щодо десктопного додатку, то мінімальними системними вимогами є такі: 2 GB RAM, Windows 7, 8, 8.1, 10 або пізніші версії.

Версія клієнту в форматі сайту не має важких обчислень на стороні клієнта,

тому варто звертати увагу лише на вимоги браузера. Розглянемо мінімльні системні характеристики необхідні для роботи з сайтом: Windows 7, Windows 8, Windows 8.1, Windows 10 або пізнішої версії. Процесор Intel Pentium 4 або більш пізніша версія з підтримкою SSE3.

2. Сервер. Мінімальні вимоги: 1 VCPU, 2 GB RAM, 20 GB DISK LOCAL

5. Конструктивні вимоги.

Система повинна бути зручною у використанні та обслуговуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

- пояснювальна записка до МКР;
- технічне завдання;
- лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз, вибір та обґрунтування актуальності розробки	20.02.2021– 8.02.21
2	Аналіз існуючих аналогів	01.03.20 – 12.03.21
3	Розробка методів і моделей системи пізнавання обличчя	13.03.21 – 22.03.21
4	Розробка макетів графічного інтерфейсу	23.03.21 – 31.03.21
5	Програмна реалізація системи	01.04.21 – 25.04.21
6	Тестування роботи системи	26.04.21 – 30.04.21

7	Економічна частина	01.05.21 – 16.05.21
8	Оформлення матеріалів до захисту МКР	17.05.21– 31.05.21

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

Додаток Б – Лістинг скрипту для побудови бази облич

```
# import the necessary packages
from imutils import paths
import face_recognition
import argparse
import pickle
import cv2
import os

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--dataset", required=True,
	help="path to input directory of faces + images")
ap.add_argument("-e", "--encodings", required=True,
	help="path to serialized db of facial encodings")
ap.add_argument("-d", "--detection-method", type=str, default="cnn",
	help="face detection model to use: either `hog` or `cnn`")
args = vars(ap.parse_args())

# grab the paths to the input images in our dataset
print("[INFO] quantifying faces...")
imagePaths = list(paths.list_images(args["dataset"]))
# initialize the list of known encodings and known names
knownEncodings = []
knownNames = []

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
```

```
        # extract the person name from the image path
        print("[INFO] processing image {}/{}".format(i + 1,
            len(imagePaths)))
        name = imagePath.split(os.path.sep)[-2]
        # load the input image and convert it from BGR (OpenCV ordering)
        # to dlib ordering (RGB)
        image = cv2.imread(imagePath)
        rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # detect the (x, y)-coordinates of the bounding boxes
        # corresponding to each face in the input image
        boxes = face_recognition.face_locations(rgb,
            model=args["detection_method"])
        # compute the facial embedding for the face
        encodings = face_recognition.face_encodings(rgb, boxes)
        # loop over the encodings
        for encoding in encodings:
            # add each encoding + name to our set of known names and
            # encodings
            knownEncodings.append(encoding)
            knownNames.append(name)
# dump the facial encodings + names to disk
print("[INFO] serializing encodings...")
data = {"encodings": knownEncodings, "names": knownNames}
f = open(args["encodings"], "wb")
f.write(pickle.dumps(data))
f.close()
```

Додаток В – Лістингскриптудля розпізнання обличчя на зображенні

```
CREATE TABLE person
 (
   person_id INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 1),
   firstname VARCHAR(255),
   lastname  VARCHAR(255),
   phone     VARCHAR(255),
   email     VARCHAR(255),
   user_id   INTEGER NOT NULL,
   PRIMARY KEY (person_id)
 );

CREATE TABLE facefeature
 (
   face_feature_id INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 1),
   feature        VARCHAR(255),
   face_id        INTEGER NOT NULL,
   PRIMARY KEY (face_feature_id)
 );

CREATE TABLE face
 (
   face_id       INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 1),
   person_id INTEGER NOT NULL,
   faceblob  LONGVARBINARY,
```

```
    facetype  VARCHAR(20),
    user_id   INTEGER NOT NULL,
    PRIMARY KEY (face_id)
  );

CREATE TABLE faceprocessqueue
  (
    face_id      INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 1),
    person_id INTEGER,
    faceblob  LONGVARBINARY,
    facetype  VARCHAR(20),
    PRIMARY KEY (face_id)
  );

CREATE TABLE "user"
  (
    user_id      INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 1),
    firstname VARCHAR(255),
    lastname  VARCHAR(255),
    phone     VARCHAR(255),
    email     VARCHAR(255),
    username  VARCHAR(255),
    PASSWORD  VARCHAR(255),
    usertype  VARCHAR(20),
    PRIMARY KEY (user_id)
  );

ALTER TABLE facefeature
```

```
        ADD CONSTRAINT fkfacefeatur702930 FOREIGN KEY (face_id)
REFERENCES face (
        face_id);

    ALTER TABLE face
        ADD CONSTRAINT fkface991538 FOREIGN KEY (person_id)
REFERENCES person (
        person_id);

    ALTER TABLE face
        ADD CONSTRAINT fkface589952 FOREIGN KEY (user_id)
REFERENCES "user" (user_id);

    ALTER TABLE person
        ADD CONSTRAINT fkperson443069 FOREIGN KEY (user_id)
REFERENCES "user" (user_id
        );

    ALTER TABLE faceprocessqueue
        ADD CONSTRAINT fkfaceproces4745 FOREIGN KEY (person_id)
REFERENCES person (
        person_id);
```

Додаток Г–Лістинг скрипту для розпізнання обличчя у відеопотоці

```
# import the necessary packages
import face_recognition
import argparse
import pickle
import cv2
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-e", "--encodings", required=True,
	help="path to serialized db of facial encodings")
ap.add_argument("-i", "--image", required=True,
	help="path to input image")
ap.add_argument("-d", "--detection-method", type=str, default="cnn",
	help="face detection model to use: either `hog` or `cnn`")
args = vars(ap.parse_args())

# load the known faces and embeddings
print("[INFO] loading encodings...")
data = pickle.loads(open(args["encodings"], "rb").read())
# load the input image and convert it from BGR to RGB
image = cv2.imread(args["image"])
rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
# detect the (x, y)-coordinates of the bounding boxes corresponding
# to each face in the input image, then compute the facial embeddings
# for each face
print("[INFO] recognizing faces...")
boxes = face_recognition.face_locations(rgb,
	model=args["detection_method"])
```

```
encodings = face_recognition.face_encodings(rgb, boxes)
# initialize the list of names for each face detected
names = []

# loop over the facial embeddings
for encoding in encodings:
    # attempt to match each face in the input image to our known
    # encodings
    matches = face_recognition.compare_faces(data["encodings"],
        encoding)
    name = "Unknown"

    # check to see if we have found a match
    if True in matches:
        # find the indexes of all matched faces then initialize a
        # dictionary to count the total number of times each face
        # was matched
        matchedIdxs = [i for (i, b) in enumerate(matches) if b]
        counts = {}
        # loop over the matched indexes and maintain a count for
        # each recognized face face
        for i in matchedIdxs:
            name = data["names"][i]
            counts[name] = counts.get(name, 0) + 1
        # determine the recognized face with the largest number of
        # votes (note: in the event of an unlikely tie Python will
        # select first entry in the dictionary)
        name = max(counts, key=counts.get)

    # update the list of names
```

```
        names.append(name)
    # loop over the recognized faces
for ((top, right, bottom, left), name) in zip(boxes, names):
        # draw the predicted face name on the image
        cv2.rectangle(image, (left, top), (right, bottom), (0, 255, 0), 2)
        y = top - 15 if top - 15 > 15 else top + 15
        cv2.putText(image, name, (left, y), cv2.FONT_HERSHEY_SIMPLEX,
            0.75, (0, 255, 0), 2)
# show the output image
cv2.imshow("Image", image)
cv2.waitKey(0)
```

Додаток Д – Лістинг команд SQL для створення бази даних

```
# import the necessary packages
from imutils.video import VideoStream
import face_recognition
import argparse
import imutils
import pickle
import time
import cv2
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-e", "--encodings", required=True,
	help="path to serialized db of facial encodings")
ap.add_argument("-o", "--output", type=str,
	help="path to output video")
ap.add_argument("-y", "--display", type=int, default=1,
	help="whether or not to display output frame to screen")
ap.add_argument("-d", "--detection-method", type=str, default="cnn",
	help="face detection model to use: either `hog` or `cnn`")
args = vars(ap.parse_args())
# load the known faces and embeddings
print("[INFO] loading encodings...")
data = pickle.loads(open(args["encodings"], "rb").read())
# initialize the video stream and pointer to output video file, then
# allow the camera sensor to warm up
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
writer = None
time.sleep(2.0)
```

```
# loop over frames from the video file stream
while True:
	# grab the frame from the threaded video stream
	frame = vs.read()

	# convert the input frame from BGR to RGB then resize it to have
	# a width of 750px (to speedup processing)
	rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
	rgb = imutils.resize(frame, width=750)
	r = frame.shape[1] / float(rgb.shape[1])
	# detect the (x, y)-coordinates of the bounding boxes
	# corresponding to each face in the input frame, then compute
	# the facial embeddings for each face
	boxes = face_recognition.face_locations(rgb,
		model=args["detection_method"])
	encodings = face_recognition.face_encodings(rgb, boxes)
	names = []
	# loop over the facial embeddings
	for encoding in encodings:
		# attempt to match each face in the input image to our known
		# encodings
		matches = face_recognition.compare_faces(data["encodings"],
			encoding)
		name = "Unknown"
		# check to see if we have found a match
		if True in matches:
			# find the indexes of all matched faces then initialize a
			# dictionary to count the total number of times each face
			# was matched
			matchedIdxs = [i for (i, b) in enumerate(matches) if b]
```

```
                counts = {}
                # loop over the matched indexes and maintain a count for
                # each recognized face face
                for i in matchedIdxs:
                    name = data["names"][i]
                    counts[name] = counts.get(name, 0) + 1
                # determine the recognized face with the largest number
                # of votes (note: in the event of an unlikely tie Python
                # will select first entry in the dictionary)
                name = max(counts, key=counts.get)

            # update the list of names
            names.append(name)
        # loop over the recognized faces
        for ((top, right, bottom, left), name) in zip(boxes, names):
            # rescale the face coordinates
            top = int(top * r)
            right = int(right * r)
            bottom = int(bottom * r)
            left = int(left * r)
            # draw the predicted face name on the image
            cv2.rectangle(frame, (left, top), (right, bottom),
                (0, 255, 0), 2)
            y = top - 15 if top - 15 > 15 else top + 15
            cv2.putText(frame, name, (left, y),
cv2.FONT_HERSHEY_SIMPLEX,
                0.75, (0, 255, 0), 2)
        # if the video writer is None *AND* we are supposed to write
        # the output video to disk initialize the writer
        if writer is None and args["output"] is not None:
```

```
                fourcc = cv2.VideoWriter_fourcc(*"MJPG")
                writer = cv2.VideoWriter(args["output"], fourcc, 20,
                        (frame.shape[1], frame.shape[0]), True)
        # if the writer is not None, write the frame with recognized
        # faces to disk
        if writer is not None:
                writer.write(frame)
        # check to see if we are supposed to display the output frame to
        # the screen
        if args["display"] > 0:
                cv2.imshow("Frame", frame)
                key = cv2.waitKey(1) & 0xFF
                # if the `q` key was pressed, break from the loop
                if key == ord("q"):
                        break
         # do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
# check to see if the video writer point needs to be released
if writer is not None:
        writer.release()
```

Додаток Е – Лістинг функцій для опрацювання облич

```
import PIL.Image
import dlib
import numpy as np
from PIL import ImageFile

try:
    import face_recognition_models
except Exception:
    print("Please install `face_recognition_models` with this command before using `face_recognition`:\n")
    print("pip install git+https://github.com/ageitgey/face_recognition_models")
    quit()

ImageFile.LOAD_TRUNCATED_IMAGES = True

face_detector = dlib.get_frontal_face_detector()

predictor_68_point_model = face_recognition_models.pose_predictor_model_location()
pose_predictor_68_point = dlib.shape_predictor(predictor_68_point_model)

predictor_5_point_model = face_recognition_models.pose_predictor_five_point_model_location()
pose_predictor_5_point = dlib.shape_predictor(predictor_5_point_model)

cnn_face_detection_model = face_recognition_models.cnn_face_detector_model_location()
cnn_face_detector = dlib.cnn_face_detection_model_v1(cnn_face_detection_model)
```

```
face_recognition_model =
face_recognition_models.face_recognition_model_location()
face_encoder = dlib.face_recognition_model_v1(face_recognition_model)



def _rect_to_css(rect):
    return rect.top(), rect.right(), rect.bottom(), rect.left()


def _css_to_rect(css):
   return dlib.rectangle(css[3], css[0], css[1], css[2])


def _trim_css_to_bounds(css, image_shape):
    return max(css[0], 0), min(css[1], image_shape[1]), min(css[2], image_shape[0]),
max(css[3], 0)


def face_distance(face_encodings, face_to_compare):
   if len(face_encodings) == 0:
        return np.empty((0))
    return np.linalg.norm(face_encodings - face_to_compare, axis=1)


def load_image_file(file, mode='RGB'):
    im = PIL.Image.open(file)
    if mode:
        im = im.convert(mode)
    return np.array(im)


def _raw_face_locations(img, number_of_times_to_upsample=1, model="hog"):
    if model == "cnn":
```

```
        return cnn_face_detector(img, number_of_times_to_upsample)
    else:
        return face_detector(img, number_of_times_to_upsample)


def face_locations(img, number_of_times_to_upsample=1, model="hog"):
    if model == "cnn":
        return [_trim_css_to_bounds(_rect_to_css(face.rect), img.shape) for face in
_raw_face_locations(img, number_of_times_to_upsample, "cnn")]
    else:
        return [_trim_css_to_bounds(_rect_to_css(face), img.shape) for face in
_raw_face_locations(img, number_of_times_to_upsample, model)]


def _raw_face_locations_batched(images, number_of_times_to_upsample=1,
batch_size=128):
    return cnn_face_detector(images, number_of_times_to_upsample,
batch_size=batch_size)


def batch_face_locations(images, number_of_times_to_upsample=1,
batch_size=128):
    def convert_cnn_detections_to_css(detections):
        return [_trim_css_to_bounds(_rect_to_css(face.rect), images[0].shape) for face
in detections]


    raw_detections_batched = _raw_face_locations_batched(images,
number_of_times_to_upsample, batch_size)


    return list(map(convert_cnn_detections_to_css, raw_detections_batched))


def _raw_face_landmarks(face_image, face_locations=None, model="large"):
    if face_locations is None:
        face_locations = _raw_face_locations(face_image)
```

```
    else:
        face_locations = [_css_to_rect(face_location) for face_location in
face_locations]

    pose_predictor = pose_predictor_68_point

    if model == "small":
        pose_predictor = pose_predictor_5_point

    return [pose_predictor(face_image, face_location) for face_location in
face_locations]


def face_landmarks(face_image, face_locations=None, model="large"):
    landmarks = _raw_face_landmarks(face_image, face_locations, model)
    landmarks_as_tuples = [[(p.x, p.y) for p in landmark.parts()] for landmark in
landmarks]

    # For a definition of each point index, see https://cdn-images-
1.medium.com/max/1600/1*AbEg31EgkbXSQehuNJBlWg.png
    if model == 'large':
        return [{
            "chin": points[0:17],
            "left_eyebrow": points[17:22],
            "right_eyebrow": points[22:27],
            "nose_bridge": points[27:31],
            "nose_tip": points[31:36],
            "left_eye": points[36:42],
            "right_eye": points[42:48],
```

```
        "top_lip": points[48:55] + [points[64]] + [points[63]] + [points[62]] +
[points[61]] + [points[60]],

        "bottom_lip": points[54:60] + [points[48]] + [points[60]] + [points[67]] +
[points[66]] + [points[65]] + [points[64]]

    } for points in landmarks_as_tuples]

  elif model == 'small':

    return [{

        "nose_tip": [points[4]],

        "left_eye": points[2:4],

        "right_eye": points[0:2],

    } for points in landmarks_as_tuples]

  else:

    raise ValueError("Invalid landmarks model type. Supported models are ['small',
'large'].")


def face_encodings(face_image, known_face_locations=None, num_jitters=1,
model="small"):

  raw_landmarks = _raw_face_landmarks(face_image, known_face_locations,
model)

  return [np.array(face_encoder.compute_face_descriptor(face_image,
raw_landmark_set, num_jitters)) for raw_landmark_set in raw_landmarks]


def compare_faces(known_face_encodings, face_encoding_to_check, tolerance=0.6):

  return list(face_distance(known_face_encodings, face_encoding_to_check) <=
tolerance)
```

Додаток Є – Лістинг класу для детекції обличчя

```
from pathlib import Path

import torch
from numpy import ndarray
from interface import implements

from ..IFaceDetector import IFaceDetector
from .vision.ssd.mb_tiny_RFB_fd import create_Mb_Tiny_RFB_fd,
create_Mb_Tiny_RFB_fd_predictor
from .vision.ssd.config.fd_config import define_img_size
from ...helpers.image_helper import image_resize

class UlfgFaceDetector(implements(IFaceDetector)):

    def __init__(self, input_img_width=640) -> None:
        define_img_size(input_img_width)

        voc_model_path = Path(__file__).parent / "./models/voc-model-labels.txt"

        class_names = [name.strip() for name in open(voc_model_path).readlines()]
        num_classes = len(class_names)
        test_device = TestDevice.CPU
        is_test = True
        self.__candidate_size = 1000
        self.__threshold = 0.7
```

```
        model_path = Path(__file__).parent / "./models/pretrained/version-RFB-640.pth"
        print("[INFO] loading ULFG face detector net...")

        net = create_Mb_Tiny_RFB_fd(num_classes, is_test=is_test, device=test_device)
        self.__predictor = create_Mb_Tiny_RFB_fd_predictor(net, candidate_size=self.__candidate_size,
                                                           device=test_device)
        net.load(model_path)

    def detect_faces(self, image: ndarray) -> list:
        boxes_tensor, labels, probs = self.__predictor.predict(image, self.__candidate_size / 2, self.__threshold)

        boxes_tensor_int = boxes_tensor.type(torch.IntTensor)
        return boxes_tensor_int

class TestDevice:
    CPU = "cpu"
    GPU = "cuda:0"

import torch

from ..utils import box_utils
from .data_preprocessing import PredictionTransform

class Predictor:
    def __init__(self, net, size, mean=0.0, std=1.0, nms_method=None,
```

```
            iou_threshold=0.3, filter_threshold=0.01, candidate_size=200, sigma=0.5,
device=None):
        self.net = net
        self.transform = PredictionTransform(size, mean, std)
        self.iou_threshold = iou_threshold
        self.filter_threshold = filter_threshold
        self.candidate_size = candidate_size
        self.nms_method = nms_method

        self.sigma = sigma
        if device:
            self.device = device
        else:
            self.device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")

        self.net.to(self.device)
        self.net.eval()

    def predict(self, image, top_k=-1, prob_threshold=None):
        cpu_device = torch.device("cpu")
        height, width, _ = image.shape
        image = self.transform(image)
        images = image.unsqueeze(0)
        images = images.to(self.device)
        with torch.no_grad():
            for i in range(1):
                scores, boxes = self.net.forward(images)
        boxes = boxes[0]
        scores = scores[0]
```

```
if not prob_threshold:
    prob_threshold = self.filter_threshold
# this version of nms is slower on GPU, so we move data to CPU.
boxes = boxes.to(cpu_device)
scores = scores.to(cpu_device)
picked_box_probs = []
picked_labels = []
for class_index in range(1, scores.size(1)):
    probs = scores[:, class_index]
    mask = probs > prob_threshold
    probs = probs[mask]
    if probs.size(0) == 0:
        continue
    subset_boxes = boxes[mask, :]
    box_probs = torch.cat([subset_boxes, probs.reshape(-1, 1)], dim=1)
    box_probs = box_utils.nms(box_probs, self.nms_method,
                              score_threshold=prob_threshold,
                              iou_threshold=self.iou_threshold,
                              sigma=self.sigma,
                              top_k=top_k,
                              candidate_size=self.candidate_size)
    picked_box_probs.append(box_probs)
    picked_labels.extend([class_index] * box_probs.size(0))
if not picked_box_probs:
    return torch.tensor([]), torch.tensor([]), torch.tensor([])
picked_box_probs = torch.cat(picked_box_probs)
picked_box_probs[:, 0] *= width
picked_box_probs[:, 1] *= height
picked_box_probs[:, 2] *= width
```

```
    picked_box_probs[:, 3] *= height
    return picked_box_probs[:, :4], torch.tensor(picked_labels), picked_box_probs[:,
4]
```

Додаток Ж – Ілюстративний матеріал до захисту магістерської кваліфікаційної роботи

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, к. т. н., доц. кафедри ПЗ _____ О. М. Хошаба

Рецензент, д.т.н., проф. кафедри КН _____ О. М. Василевський

Нормоконтроль, к. т. н., доцент кафедри ПЗ _____ О. М.Хошаба

Виконавець, студент групи 1ПІ–19м _____С. С. Штокал

Рисунок Е.1 – Титулка

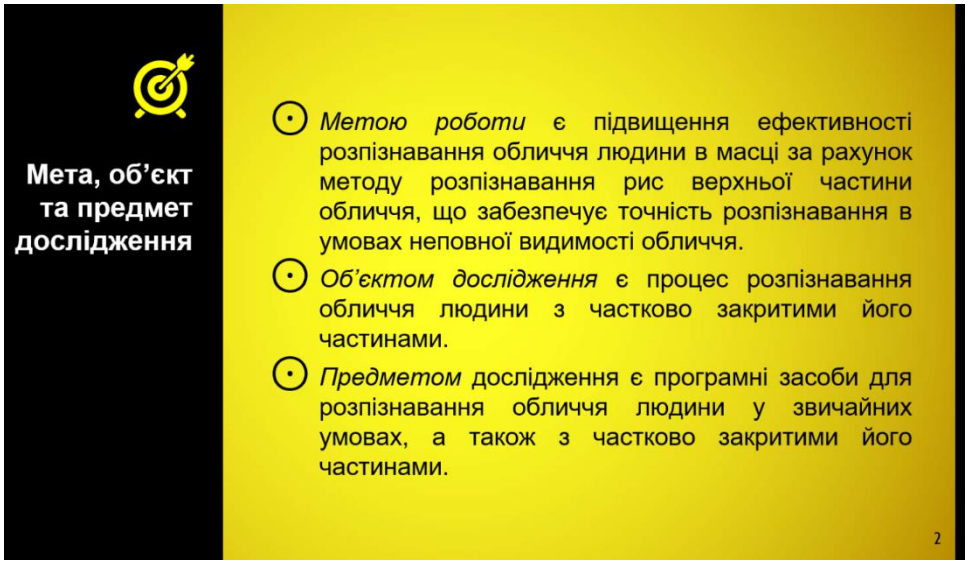


Рисунок Е.2 – Мета, об'єкт та предмет дослідження

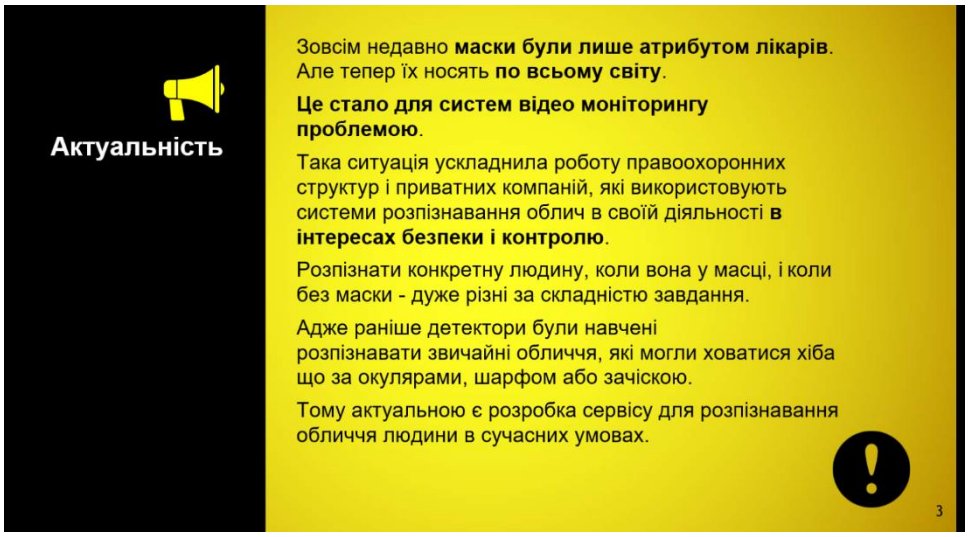


Рисунок Е.3 – Актуальність дослідження

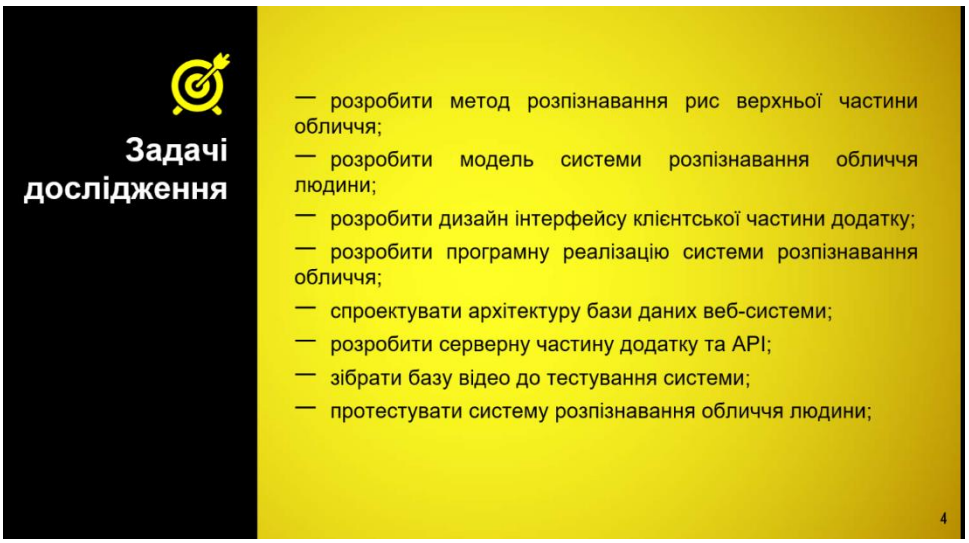


Рисунок Е.4 – Задачі дослідження

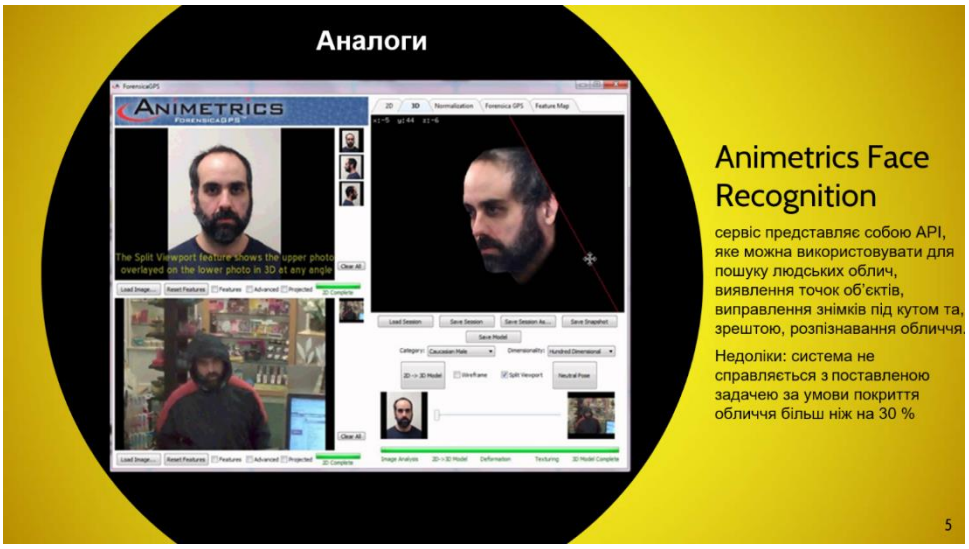


Рисунок Е.5 – Аналог «Animetrics Face Recognition »

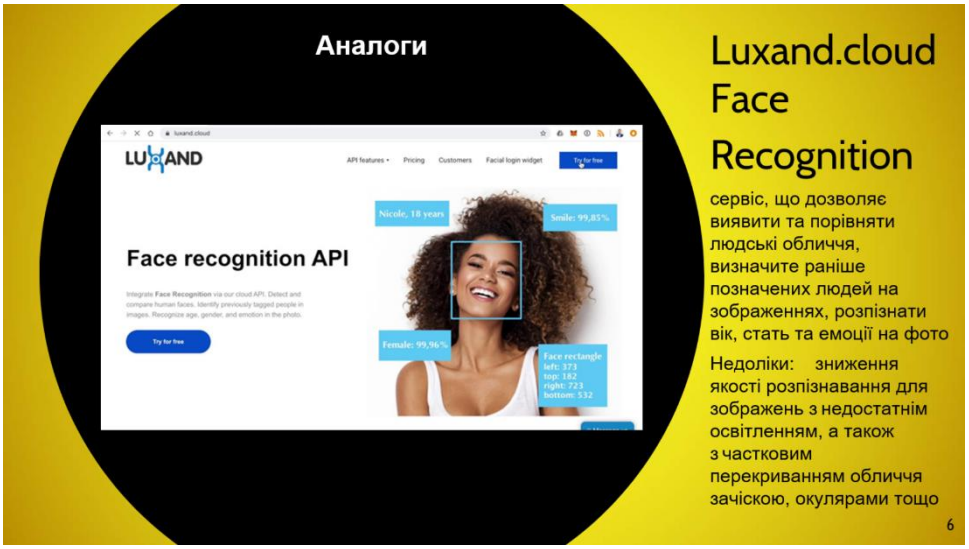


Рисунок Е.6 – Аналог « Luxand.cloud Face Recognition »

Рисунок Е.7 – Аналог « SenseTime »

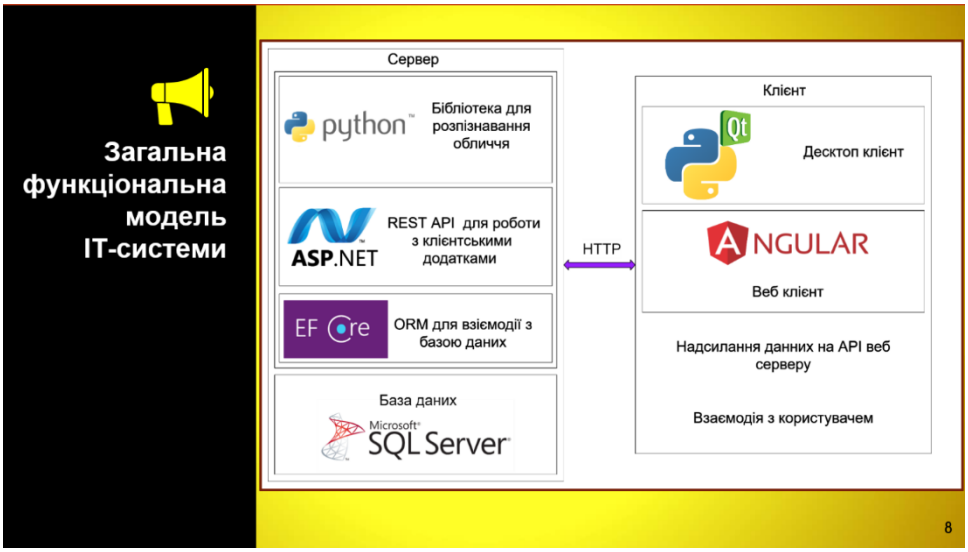


Рисунок Е.8 – Загальна функціональна модель IT-системи

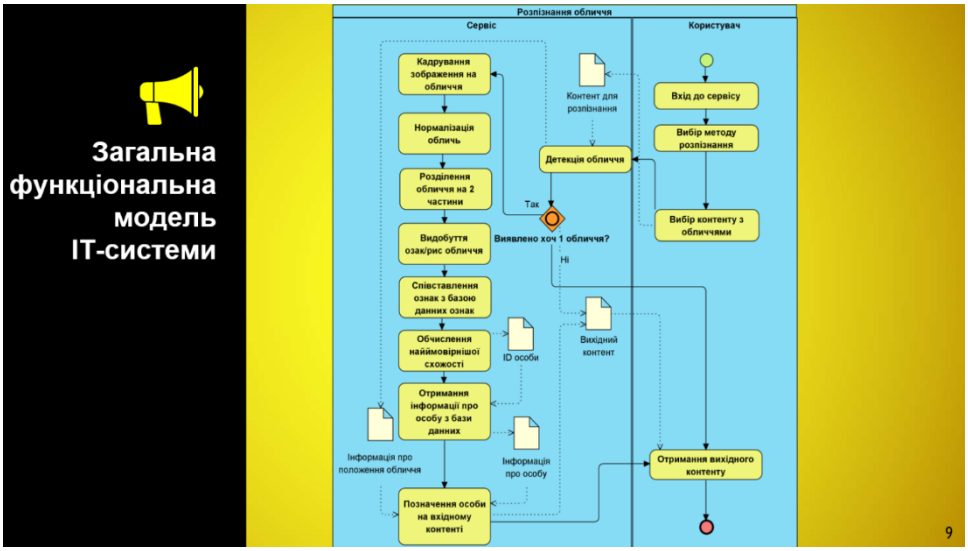


Рисунок Е.9 – Схема процесу розпізнання обличчя

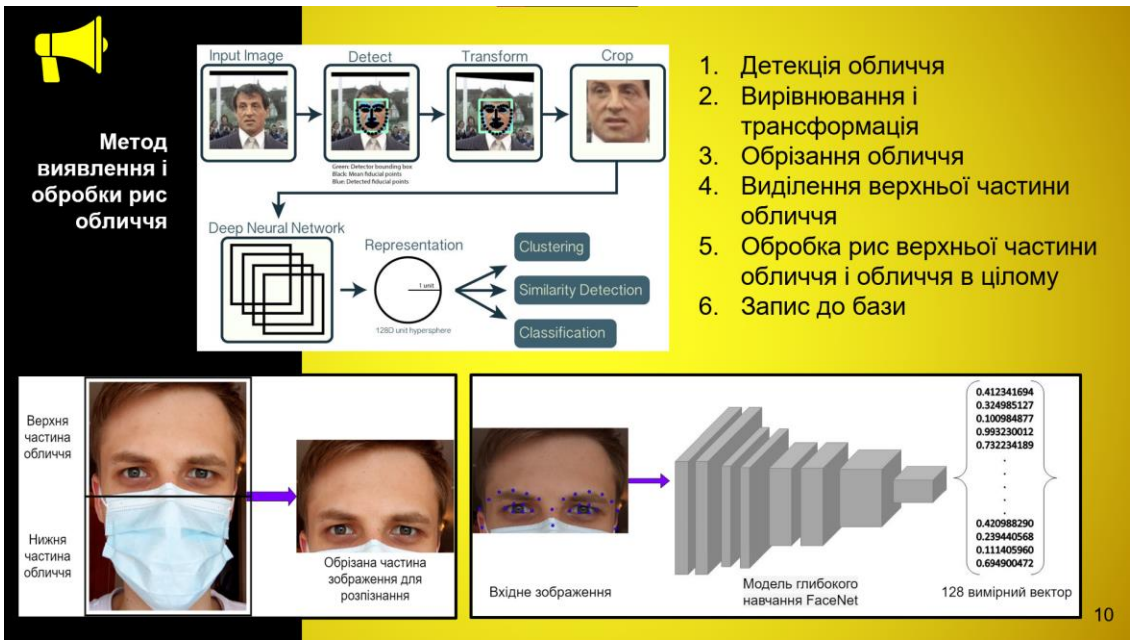
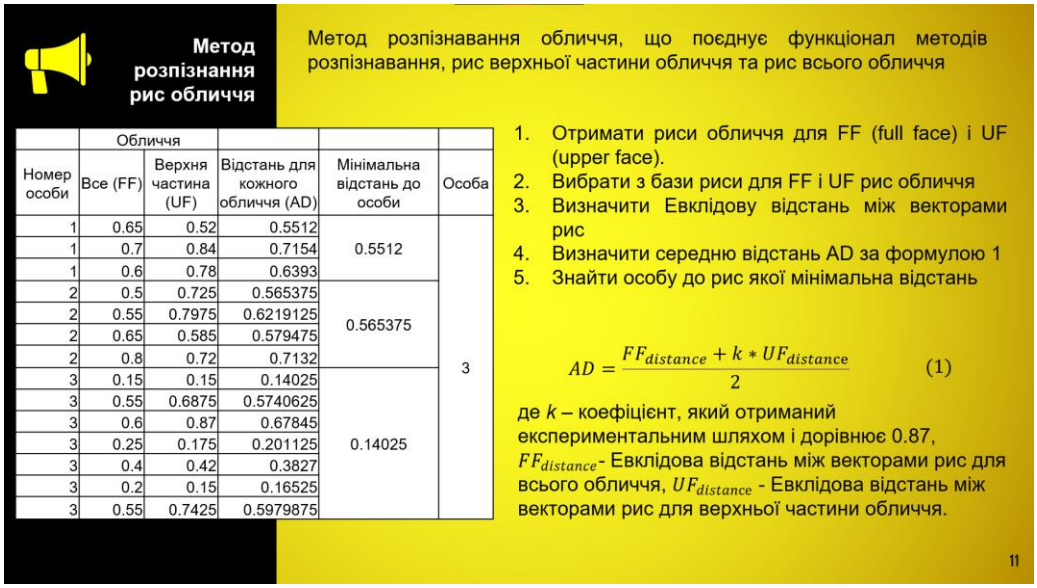


Рисунок Е.10 – Методвиявлення і обробки рис обличчя



	Обличчя				
Номер особи	Все (FF)	Верхня частина (UF)	Відстань для кожного обличчя (AD)	Мінімальна відстань до особи	Особа
1	0.65	0.52	0.5512	0.5512	3
1	0.7	0.84	0.7154		
1	0.6	0.78	0.6393		
2	0.5	0.725	0.565375	0.565375	
2	0.55	0.7975	0.6219125		
2	0.65	0.585	0.579475		
2	0.8	0.72	0.7132		
3	0.15	0.15	0.14025	0.14025	
3	0.55	0.6875	0.5740625		
3	0.6	0.87	0.67845		
3	0.25	0.175	0.201125		
3	0.4	0.42	0.3827		
3	0.2	0.15	0.16525		
3	0.55	0.7425	0.5979875		

Рисунок Е.11 – Метод розпізнавання рис обличчя

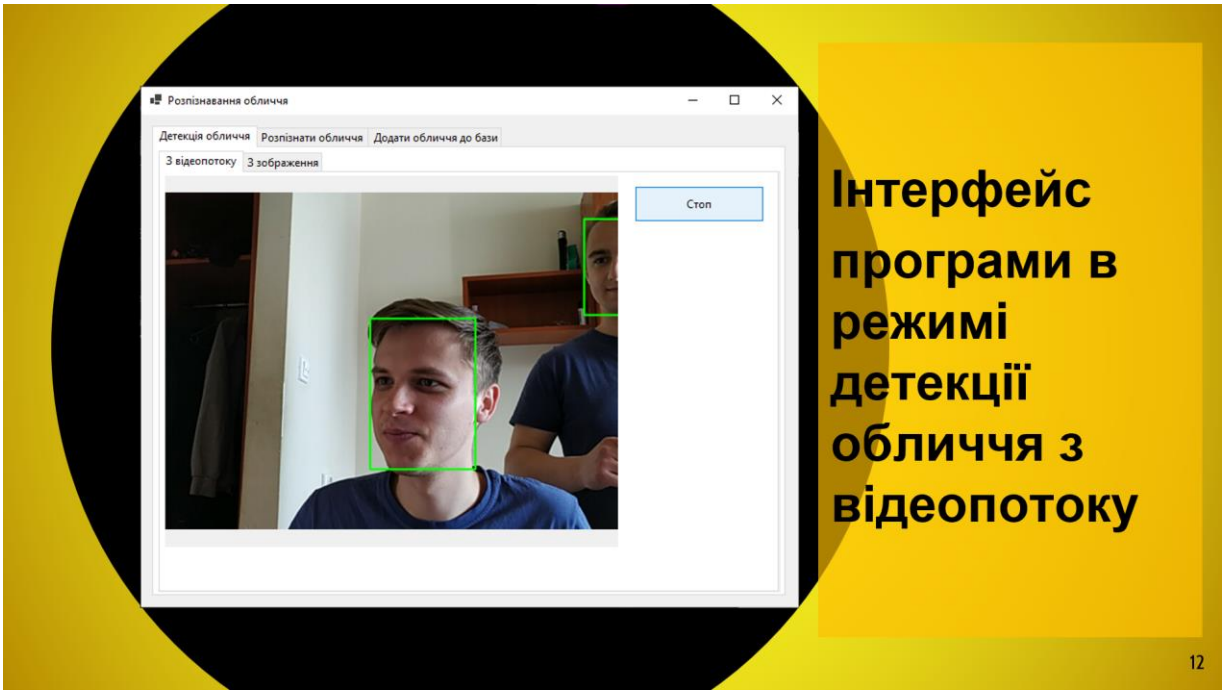


Рисунок Е.12 – Інтерфейспрограми в режимі детекції обличчя з відеопотоку

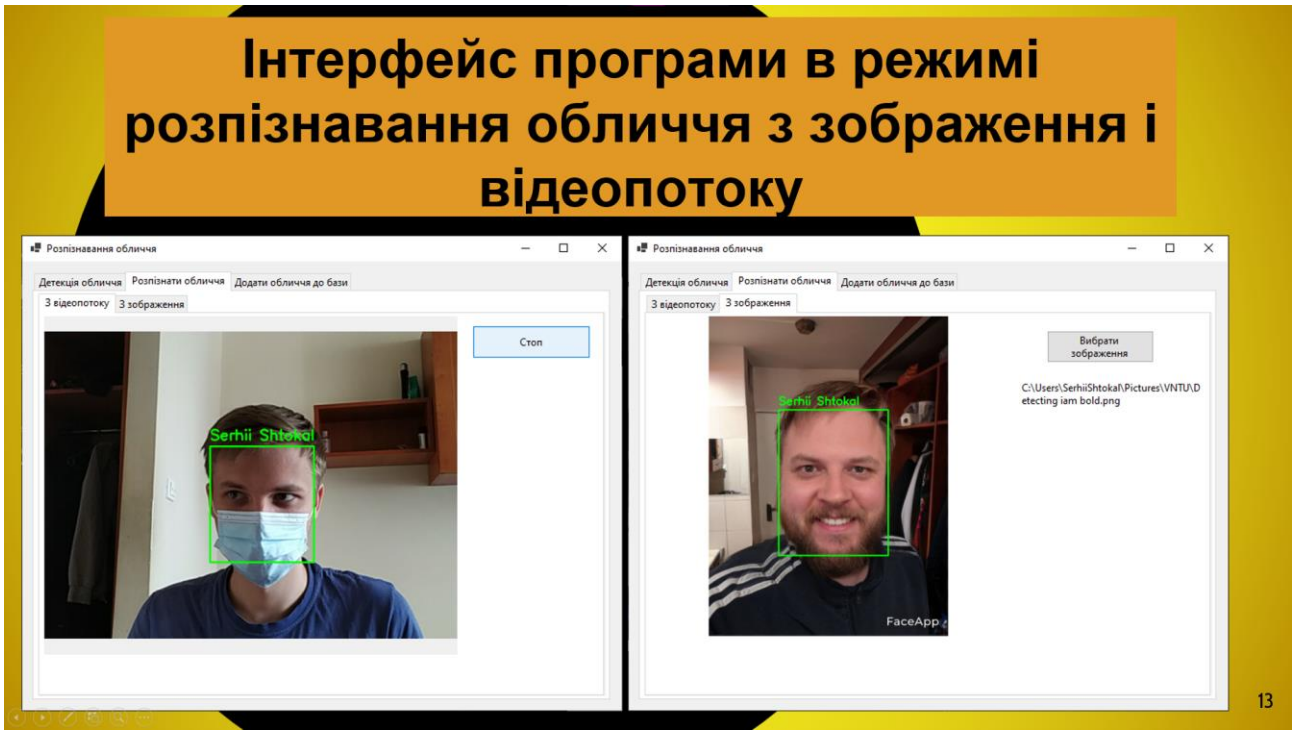


Рисунок Е.13 – Інтерфейс програми в режимі розпізнавання обличчя з зображення і відеопотоку

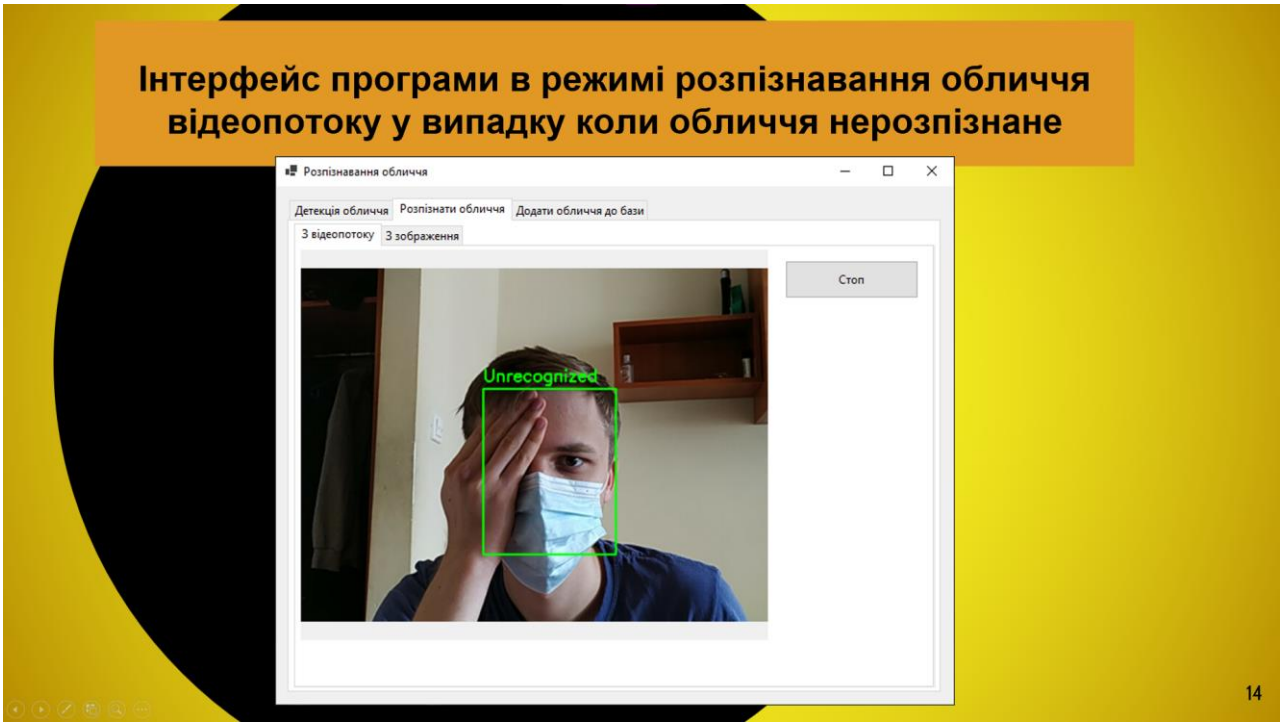


Рисунок Е.14 – Інтерфейс програми в режимі розпізнавання обличчя відеопотоку у випадку коли обличчя нерозпізнане

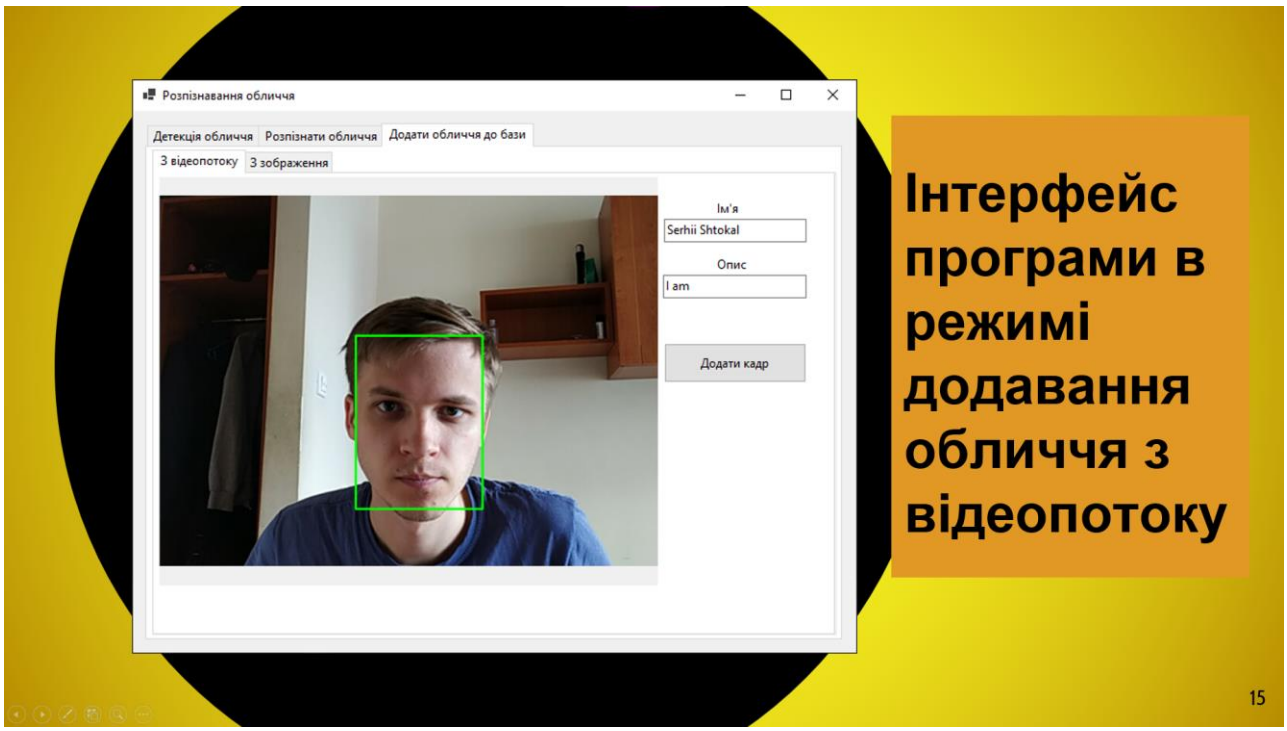


Рисунок Е.15 – Інтерфейс програми в режимі додавання обличчя з відеопотоку

Рисунок Е.16 – Відео з розпізнанням облич

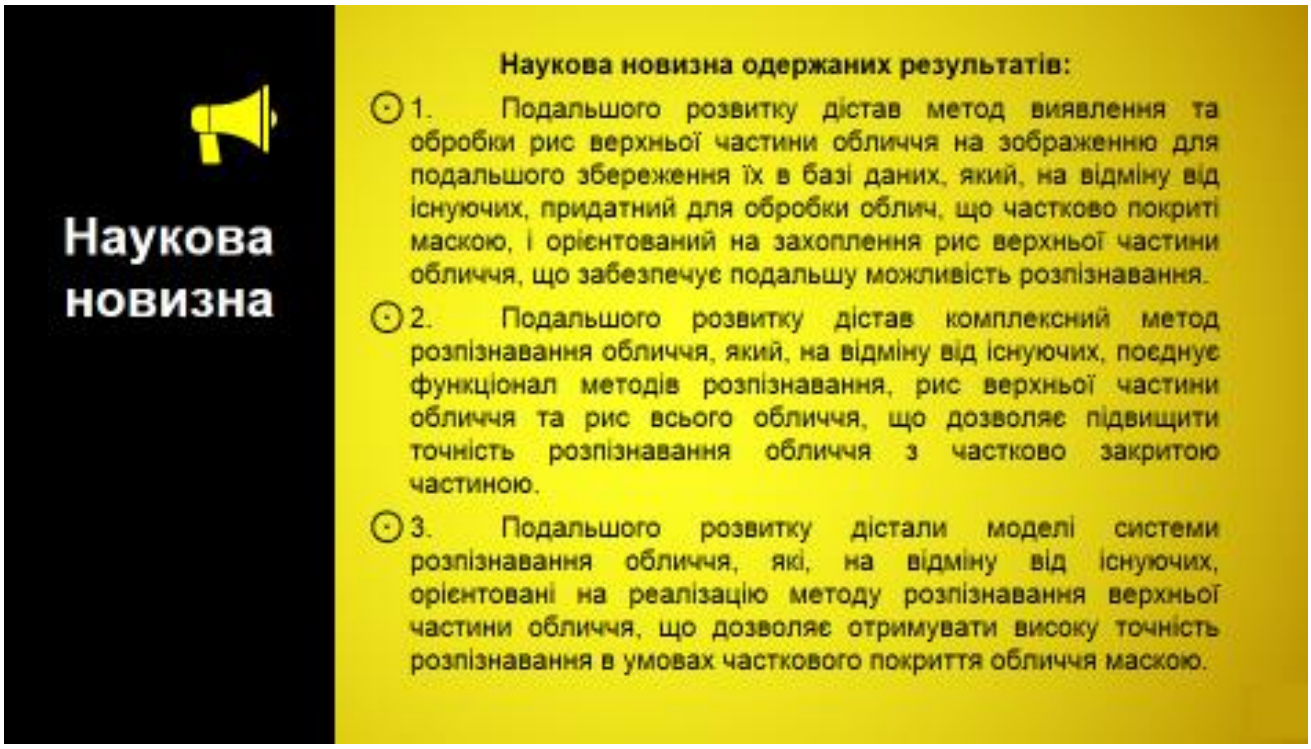


Рисунок Е.17 – Наукова новизна

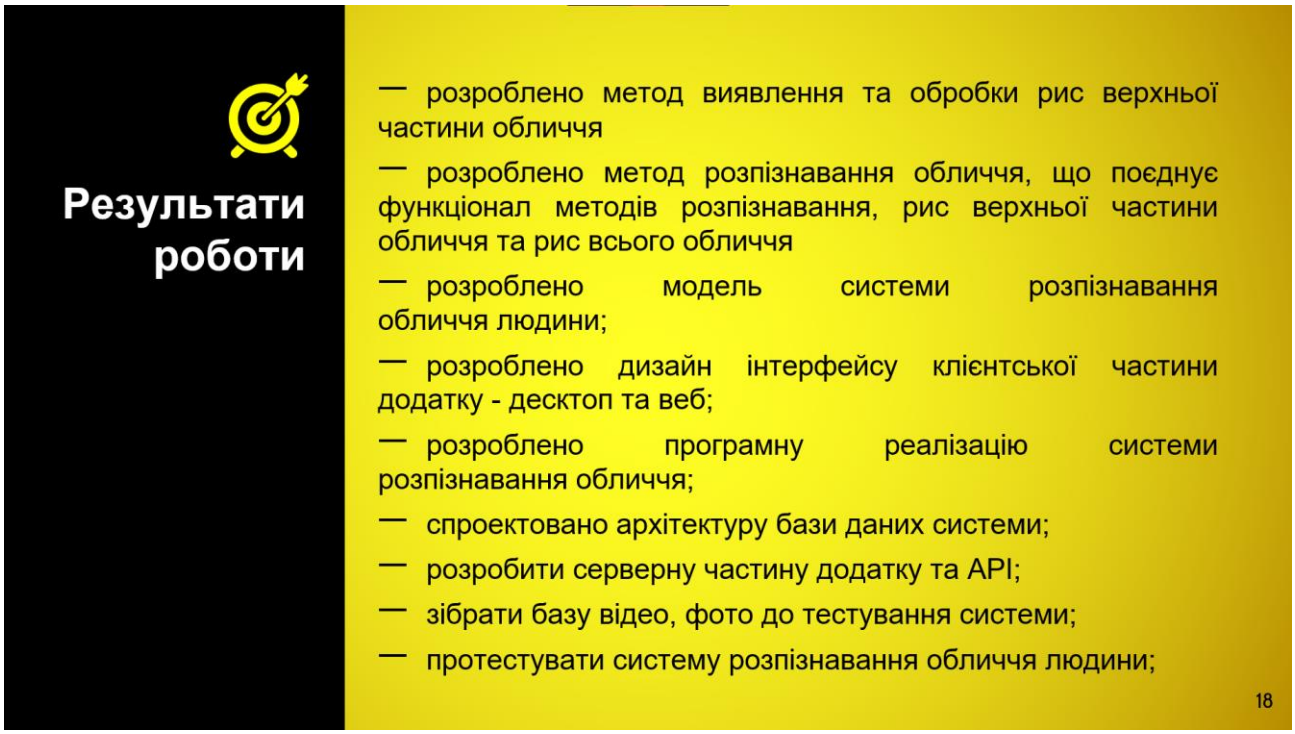


Рисунок Е.18 – Результати роботи

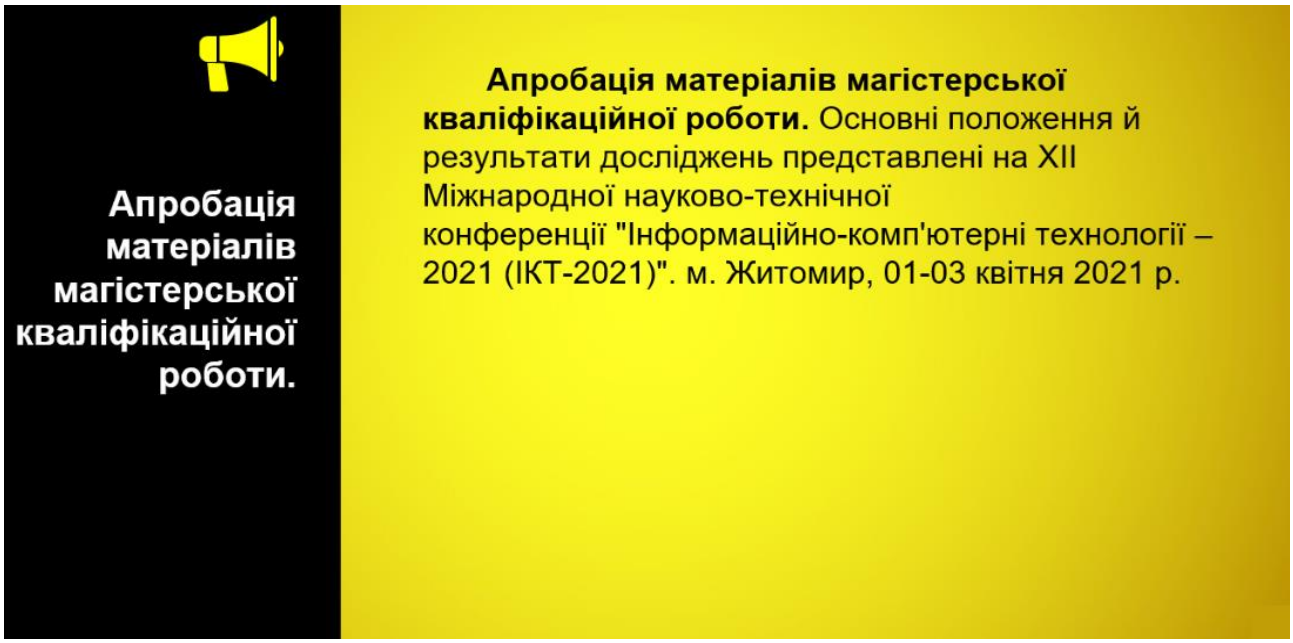


Рисунок Е.19 – Апробація матеріалів

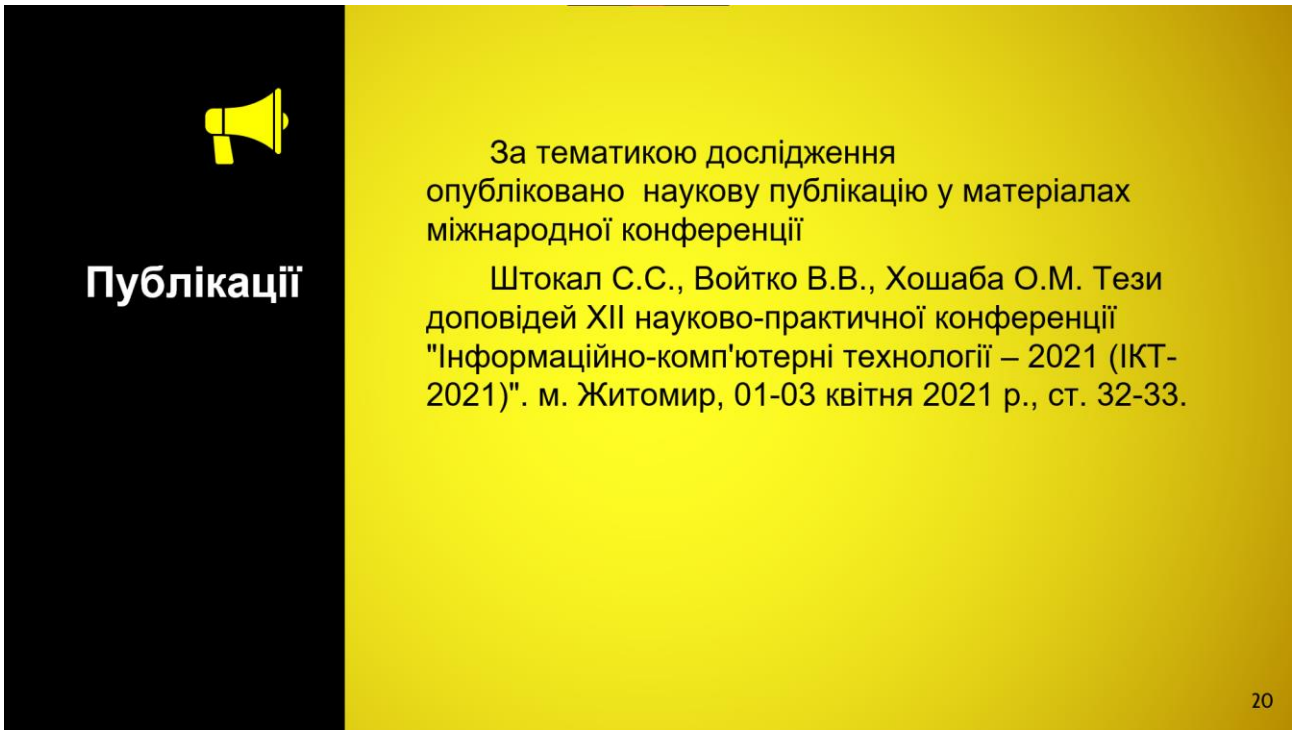


Рисунок Е.20 – Публікації