МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
Факультет мехатроніки та комп'ютерних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПОШУКУ ВІЙСЬКОВИХ ЗНИКЛИХ БЕЗВІСТИ

Рівень вищої освіти другий (магістерський)
Спеціальність 122 Комп'ютерні науки
Освітня програма Комп'ютерні науки

Виконав: студент групи МгІТ-1-23
Павлюченко Д.М.
Науковий керівник к.т.н., доцент Корогод Г.О.
Рецензент к.т.н., доцент Яхно В.М.

Київ 2024

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет Мехатроніки та комп'ютерних технологій

Кафедра Комп'ютерних наук

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувачка кафедри КН

_____ Наталія ЧУПРИНКА

«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Павлюченко Дмитру Михайловичу

1. Тема роботи: Розроблення програмного забезпечення для пошуку військових зниклих безвісти.
Науковий керівник роботи: Корогод Ганна Олександрівна, к.т.н., доцент, затверджені наказом КНУТД від "03" вересня 2024 року №188-уч.
2. Вихідні дані до кваліфікаційної роботи: Розробки кафедри комп'ютерних наук. Програмне забезпечення для пошуку військових зниклих безвісти.
3. Зміст кваліфікаційної роботи: Вступ; Розділ 1. Огляд методів та технологій розпізнання облич у задачах пошуку осіб; Розділ 2. Алгоритмічне забезпечення системи пошуку військових зниклих безвісти; Розділ 3. Програмна реалізація системи пошуку військових зниклих безвісти; Розділ 4. Інтерфейс користувача та функціональні можливості системи; Висновки; Список використаних джерел; Додатки.
4. Дата видачі завдання: _____.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ		
2	Розділ 1. Огляд методів та технологій розпізнання облич у задачах пошуку осіб		
3	Розділ 2. Алгоритмічне забезпечення системи пошуку військових зниклих безвісти		
4	Розділ 3. Програмна реалізація системи пошуку військових зниклих безвісти		
	Розділ 4. Інтерфейс користувача та функціональні можливості системи		
	Висновки		
	Оформлення кваліфікаційної магістерської роботи		
	Подача кваліфікаційної роботи науковому керівнику для відгуку		
	Подача кваліфікаційної роботи для рецензування (за 12 днів до захисту)		
	Перевірка кваліфікаційної роботи на наявність ознак плагіату та текстових співпадінь (за 10 днів до захисту)		
	Подання кваліфікаційної роботи на завідувачу кафедри		

З завданням ознайомлений:

Студент _____ Дмитро ПАВЛЮЧЕНКО

Науковий керівник _____ Ганна КОРОГОД

АНОТАЦІЯ

Павлюченко Дмитро Михайлович. Розроблення програмного забезпечення для пошуку військових зниклих безвісти.

Кваліфікаційна магістерська робота за спеціальністю 122 – «Комп'ютерні науки» – Київський національний університет технологій та дизайну, Київ, 2024 рік.

У магістерській роботі розроблено програмне забезпечення для автоматизованого пошуку військових, що зникли безвісти. В основі системи лежить використання сучасних технологій розпізнавання облич та обробки зображень на мові програмування Python. Реалізовано алгоритми виявлення, розпізнавання та порівняння облич з високою точністю (>90%) та швидкодією обробки (до 5 секунд на одне зображення).

Розроблено зручний користувацький інтерфейс у вигляді Telegram-бота, що забезпечує простоту взаємодії та відсутність додаткових затрат на написання дизайну. Впроваджено механізми забезпечення безпеки даних, включаючи шифрування персональної інформації та систему контролю доступу. Програмне забезпечення розроблено з використанням найсучасніших технологій (Python, PostgreSQL, face_recognition, OpenCV) з можливістю масштабування.

Ключові слова: розпізнавання облич, пошук зниклих безвісти, telegram bot, глибоке навчання, моніторинг соціальних мереж.

ABSTRACTS

Pavlyuchenko Dmytro Mykhailovych. Development of software for searching for missing military personnel.

Qualification master's thesis in specialty 122 - "Computer Science" - Kyiv National University of Technologies and Design, Kyiv, 2024.

In the master's thesis, software for automated search for missing military personnel has been developed. The system is based on the use of modern technologies for face recognition and image processing in the Python programming language. Algorithms for detecting, recognizing and comparing faces with high accuracy (>90%) and processing speed (up to 5 seconds per image) have been implemented.

A convenient user interface in the form of a Telegram bot has been developed, which ensures ease of interaction and the absence of additional costs for writing a design. Data security mechanisms have been implemented, including encryption of personal information and an access control system. The software has been developed using the most modern technologies (Python, PostgreSQL, face_recognition, OpenCV) with the ability to scale.

Keywords: face recognition, search for missing persons, telegram bot, deep learning, social media monitoring.

ЗМІСТ

ВСТУП

Проблема пошуку військовослужбовців, зниклих безвісти, є однією з найгостріших гуманітарних проблем України на поточний момент. В умовах військової агресії російської федерації проти України оперативний пошук та дистанційна, автоматична ідентифікація зниклих військовослужбовців стає максимально критично важливим завданням, що потребує ефективних технологічних рішень. Розвиток комп'ютерних технологій та штучного інтелекту (AI) відкриває нові можливості для процесу пошуку зниклих осіб: онлайн моніторинг соціальних мереж, автоматичне розпізнавання облич, співставлення з еталонними фото зниклого безвісти та інформування про визначення схожості. В умовах збільшення кількості інформації в соціальних мережах, наявності «важкого» медіаконтенту (18+), автоматизація процесу моніторингу та аналізу даного контенту стає критично важливою для психологічної безпеки людини та ефективності пошукових операцій і є надзвичайно актуальною сьогодні.

Таким чином, створення ефективних інструментів для пошуку військових, що зникли безвісти, з використанням сучасних технологій розпізнавання облич та аналізу даних є надзвичайно актуальним.

Мета дослідження полягає у розробці програмного забезпечення для автоматизованого пошуку військових, зниклих безвісти, шляхом аналізу та моніторингу соціальних мереж та месенджерів з використанням технологій розпізнавання облич з подальшим інформуванням при виявленні схожості.

Для досягнення поставленої мети дослідження необхідно вирішити наступні задачі:

1. Виконати аналіз вже існуючих методів та технологій розпізнавання облич, визначити їх переваги та обмеження.

2. Розробити архітектуру для системи автоматизованого пошуку зниклих військових.
3. Реалізувати алгоритми розпізнавання та порівняння облич.
4. Створити систему моніторингу та аналізу медіа в Telegram-каналах.
5. Розробити користувацький інтерфейс для додавання еталонних фото для пошуку, управління пошуком та систему сповіщень на основі Telegram бота.
6. Провести тестування та оптимізацію розробленого програмного забезпечення.

Об'єкт дослідження – процес автоматизованого пошуку та ідентифікації осіб на фото та відео.

Предмет дослідження – методи, алгоритми, технології розпізнавання облич в задачі пошуку зниклих безвісти військовослужбовців.

Методи дослідження – в роботі використано методи машинного навчання, обробки зображень, а також методи проектування програмних систем. Для реалізації системи застосовано сучасні технології глибокого навчання та нейронних мереж.

Наукова новизна отриманих результатів полягає в:

1. Застосування технологій розпізнавання обличь по медіаконтенту в військовому напрямку.

2. Удосконаленні методів розпізнавання облич для роботи з зображеннями різної якості з соціальних мереж та месенджерів. Розробці комплексного підходу до автоматизації процесу пошуку зниклих осіб через аналіз медіаконтенту.

3. Створенні адаптивних алгоритмів співставлення, порівняння облич з урахуванням специфіки військових фотографій, невисокої якості зображень, дублювання контенту в каналах.

Практичне значення отриманих результатів:

- Розроблено програмне забезпечення для автоматизованого пошуку зниклих військовослужбовців по медіаконтенту.
- Створено систему моніторингу соціальних мереж та месенджерів для виявлення інформації про зниклих осіб. Реалізовано механізми онлайн сповіщення про вірогідність збігу.
- Забезпечено можливість горизонтального та вертикального масштабування системи для обробки великих обсягів даних.

Результати роботи можуть бути використані рідними зниклих безвісти, волонтерськими організаціями та державними установами. Дані які обробляються – є публічними, доступними в соціальних мережах та месенджерах. Система має потенціал для подальшого розвитку та адаптації до інших задач пошуку та ідентифікації осіб в після воєнний час (пошук зниклих цивільних громадян України, зниклих дітей по відкритій інформації з медіа контентом в соціальних мережах, що є завжди актуальним).

Особистий внесок здобувача полягає в:

- Розробці архітектури програмного забезпечення.
- Реалізації алгоритмів розпізнавання та порівняння облич.
- Створенні системи моніторингу Telegram-каналів з можливістю додавання нових джерел.
- Розробці користувацького інтерфейсу, системи управління та сповіщень.
- Проведенні тестування та оптимізації системи.

Розроблене програмне забезпечення надає можливість автоматичного пошуку військових, зниклих безвісти, за рахунок автоматизації процесів моніторингу та аналізу інформації з соціальних мереж та месенджерів. Система забезпечує високу точність розпізнавання облич та онлайн

сповіщення про можливі збіги, що є критично важливим для населення України.

Апробація. Результати дослідження, що були направлені на аналіз сучасних підходів до персоналізації навчання з використанням EdTech, доповідались на VIII міжнародній науково-практичній конференції «MSIE-2024».

РОЗДІЛ 1. ОГЛЯД МЕТОДІВ ТА ТЕХНОЛОГІЙ РОЗПІЗНАВАННЯ ОБЛИЧ У ЗАДАЧАХ ПОШУКУ ОСІБ

1.1. Аналіз сучасного стану проблеми розпізнавання обличь

1.1.1. Еволюція технологій розпізнавання обличь

Розвиток технологій розпізнавання облич пройшов декілька ключових етапів [6]:

1. Геометричний підхід (1960-1970-ті роки) є перший серед всіх метод визначення обличчя. В основі методу лежить:
    - Вимірювання відстаней між ключовими точками обличчя
    - Аналіз пропорцій та форматування набору ознак

Даний метод має обмежену точність через примітивність методів та високі вимоги до освітлення.

2. Голістичний підхід (1980-1990-ті роки):
    - Eigenfaces - метод на основі аналізу головних компонент PCA (найбільш значущої інформації) для розпізнавання облич.
    - Fisherfaces - метод на основі лінійного дискримінантного аналізу LDA (для класифікації та зменшення розмірності)

3. Локальні ознаки (2000-2010 роки) – підхід у розпізнанні об’єктів, включаючи обличчя та аналізі фрагментів. Основними методами даного підходу:
    - SIFT (Scale-Invariant Feature Transform) - виявлення ключових точок, які є статичними до масштабування, обертання та зміни освітлення. Використовується для опису локальних ознак зображення.
    - SURF (Speeded Up Robust Features) - швидша альтернатива SIFT, яка забезпечує схожу точність і статичність до змін у масштабі й

освітленні, але зменшує час обчислень завдяки використанню інтегральних зображень.

- LBP (Local Binary Patterns) - техніка аналізу текстури, яка представляє пікселі в локальному патерні за допомогою двійкових значень.
- HOG (Histogram of Oriented Gradients) - метод, який аналізує напрямки градієнтів у локальних областях зображення, що дозволило ефективно розпізнавати форми та об'єкти, включаючи обличчя.

Даний підхід заклав основу для розвитку існуючих зараз систем розпізнавань обличчя. Всі сучасні методи частково використовують локальні ознаки в своїх дослідженнях для побудови нових алгоритмів.

4. Глибоке навчання «Deep Learning» (2012-теперішній час) на основі багатошарових нейронних мереж, які здатні автоматично аналізувати великі масиви даних з імітацією роботи людського мозку. Основна ідея підходу в створенні моделей, які самостійно вчяться без необхідності інженерно виділяти об’єкти. Основні методи цього підходу:

- Згорткові нейронні мережі (Convolutional Neural Networks, CNNs) - глибокі моделі, що використовуються для автоматичного видізначення ознак із зображень. Вони ефективно розпізнають обличчя завдяки можливості враховувати просторову ієрархію ознак.
- Метричне навчання (Metric Learning) - метод навчання, який оптимізує функцію подібності між парами зображень. Часто використовується в задачах перевірки облич або виявлення відповідностей.

- Ансамблеві методи (Ensemble Methods) – комбінація декількох методів для максимально ефективного визначення облич.
- Самонавчання (Self-supervised Learning) - метод, який використовує невеликі мітки даних або зовсім немарковані дані для навчання моделей. Це знижує потребу в великих наборах анотованих зображень.

1.1.2. Основні метрики оцінки якості розпізнавання

Для оцінки ефективності систем розпізнавання облич використовуються наступні метрики [11]:

1. Точність верифікації - це метрики, які використовуються для оцінки того, наскільки точно система розпізнає обличчя в задачах верифікації (визначення, чи два зображення належать одній особі):
   - False Acceptance Rate (FAR) – частка хибних визначень, що показує, як часто система помилково ідентифікує двох різних людей як одну особу.
   - False Rejection Rate (FRR) - частка хибних відмов, що визначає, як часто система помилково відмовляє у верифікації однієї й тієї самої особи.
   - Equal Error Rate (EER) - точка, в якій FAR дорівнює FRR. EER використовується як універсальний показник для оцінки загальної ефективності системи. Чим менше значення EER, тим більш збалансованою і точною є система.
   - Area Under Curve (AUC) - площа під кривою ROC (Receiver Operating Characteristic). AUC показує загальну продуктивність системи: чим ближче AUC до 1, тим краща якість системи.
2. Точність ідентифікації. Метрики, пов’язані з точністю ідентифікації, оцінюють, наскільки добре система розпізнає обличчя, коли потрібно

визначити особу серед кількох можливих кандидатів. Ці метрики використовуються для оцінки продуктивності в задачах ідентифікації:

- Rank-1 Recognition Rate - це частка випадків, коли правильний результат (особа) знаходиться на першій позиції списку кандидатів, повернутого системою. Вказує на здатність системи однозначно ідентифікувати особу з першої спроби. Чим вища Rank-1 точність, тим ефективніше система розпізнає обличчя в простих умовах.
- Cumulative Match Characteristic (CMC) - це крива, яка показує ймовірність того, що правильна особа з'явиться серед перших k позицій списку кандидатів. Оцінює здатність системи надавати кілька варіантів ідентифікації, якщо точний результат не опиняється на першій позиції.
- Mean Average Precision (mAP) - середнє значення точності на всіх рангах у вибірці даних. Використовується в задачах, де потрібно враховувати всі релевантні результати (наприклад, у задачах пошуку схожих облич). Чим вище mAP, тим краще система враховує всі релевантні варіанти ідентифікації.

3. Швидкісні характеристики. Метрики що оцінюють продуктивність системи розпізнавання обличчя в залежності від швидкості опрацювання:

- Час визначення (детекції) обличчя - час, потрібний для виявлення обличчя на зображенні або в потоці відео. Важливо для систем, які працюють в режимі реального часу таких як відео. Залежить від кількості осіб в кадрі.
- Час виділення ознак - час, потрібний для витягнення ознак обличчя після його виявлення. Чим складніший алгоритм і більша база даних ознак, тим довше триває обробка.

- Час порівняння – час, потрібний для порівняння витягнутих ознак обличчя. Критично для систем із великою кількістю користувачів (наприклад, паспортний контроль або бази даних правоохоронних органів).
- Загальний час обробки - сумарний час виконання всіх етапів (визначення обличчя, виділення ознак, порівняння).

1.2. Технології обробки та аналізу зображень

1.2.1. Попередня обробка зображень

Ключові етапи попередньої обробки включають [22]:

1. Нормалізація розміру: масштабування, обрізка, заповнення, збереження пропорцій
2. Корекція освітлення: гістограмна еквалізація, адаптивна еквалізація, гамма-корекція, локальна нормалізація
3. Фільтрація шумів: медіанний фільтр, гаусівський фільтр, білатеральний фільтр, нелокальне усереднення
4. Покращення якості: підвищення різкості, контрастування, колірна корекція, super-resolution

1.2.2. Алгоритми детекції облич

Сучасні методи детекції облич можна розділити на такі категорії [7]:

1. Каскадні детектори: Viola-Jones, LBP каскади, ACF детектори мають високу швидкість детекції
2. CNN-based детектори: MTCNN, RetinaFace, DSFD мають висока точність але потребують більших ресурсів та мають меншу швидкість обробки
3. Якірні детектори: SSD, YOLO, FPN надають баланс швидкості та точності, при середніх витратах ресурсів.

4. Гібридні підходи: каскад + CNN, двоетапна детекція, ансамблеві методи використовуються при потребі максимальної точності та необмежених ресурсах.

1.2.3. Вирівнювання та нормалізація облич

Процес вирівнювання включає наступні етапи [8]:

1. Виявлення ключових точок: очі, ніс, рот, контури обличчя
2. Геометричні перетворення: афінні перетворення, проективні перетворення, тонкі сплайни, piece-wise affine
3. Нормалізація: стандартизація положення очей, вирівнювання орієнтації, масштабування, обрізка зайвих областей
4. Верифікація якості: оцінка симетрії, перевірка пропорцій, контроль спотворень надає відбракування неякісних результатів

1.3. Аналіз існуючих рішень

1.3.1. Комерційні системи

Сучасний ринок систем розпізнавання облич представлений широким спектром комерційних рішень, які пропонують різноманітні підходи та технології для вирішення задач ідентифікації та верифікації особистості. Розглянемо детально основні комерційні системи та їх особливості:

1. Amazon Rekognition: Система, розроблена Amazon Web Services, представляє собою потужне хмарне рішення для розпізнавання облич. Основною перевагою є висока точність розпізнавання, яка досягається завдяки використанню передових нейромережевих архітектур та постійному навчанню на великих наборах даних. Система забезпечує:
    - Точність розпізнавання на рівні 99.9% для верифікації особи
    - Можливість обробки мільйонів зображень за хвилину
    - Автоматичне масштабування обчислювальних ресурсів
    - Глибоку інтеграцію з іншими сервісами AWS

- Підтримку розпізнавання в режимі реального часу
- Функції аналізу емоцій та атрибутів обличчя

2. Microsoft Azure Face: Платформа від Microsoft надає комплексне рішення для розпізнавання облич, орієнтоване на корпоративний сектор. Система базується на глибоких згорткових нейронних мережах [3] та використовує передові методи машинного навчання. Ключові особливості включають:
   - Розширений API з підтримкою різних сценаріїв використання
   - Інтеграцію з Active Directory для керування доступом
   - Відповідність міжнародним стандартам безпеки
   - Глобальну мережу дата-центрів для забезпечення низької латентності
   - Можливість налаштування порогів впевненості
   - Підтримку групового розпізнавання
3. Google Cloud Vision: Рішення від Google відрізняється глибокою оптимізацією алгоритмів та використанням власних TPU (Tensor Processing Units) для прискорення обчислень. Система пропонує:
   - Використання передових ML-моделей [13]
   - Автоматичну оптимізацію продуктивності
   - Інтеграцію з іншими сервісами Google Cloud
   - Розширені можливості аналітики та звітності
   - Підтримку Edge ML для локальної обробки
   - Можливість кастомізації моделей
4. Face++: Китайська система розпізнавання облич, яка спеціалізується на роботі з азійськими фенотипами. Використовує власні алгоритми глибокого навчання [2] та пропонує:
   - Спеціалізовані алгоритми для різних етнічних груп

- Оптимізовані мобільні SDK
- Гнучке REST API з широким функціоналом
- Можливість локального розгортання
- Високу точність при роботі з масками
- Аналіз віку та емоцій

5. IBM Watson Visual Recognition: Система від IBM відрізняється глибокою інтеграцією з екосистемою Watson та використанням передових методів штучного інтелекту. Особливості включають:
   - Можливість створення власних класифікаторів
   - Інтеграцію з Watson AI Services
   - Підтримку контейнерного розгортання [27]
   - Розширені можливості аналітики
   - Автоматичну оптимізацію моделей
   - Підтримку федеративного навчання
6. NtechLab FindFace: Російська система, що спеціалізується на відеоаналітиці та розпізнаванні облич у реальному часі. Система забезпечує:
   - Обробку відеопотоку в реальному часі
   - Високу швидкість ідентифікації
   - Підтримку edge-обчислень
   - Інтеграцію з системами відеоспостереження
   - Аналіз поведінки та траєкторій
   - Розпізнавання в складних умовах освітлення
7. Kairos Face Recognition: Система орієнтована на бізнес-застосування та забезпечує:
   - Аналіз демографічних даних
   - Вимірювання емоційної реакції

- Інтеграцію з CRM системами
- Підтримку різних платформ
- Аналітику поведінки клієнтів
- Захист від підробок

1.3.2. Відкриті рішення

Сектор відкритих рішень для розпізнавання облич представлений широким спектром бібліотек та фреймворків, які забезпечують різні рівні абстракції та можливості:

1. face_recognition [15]: Популярна Python бібліотека, що надає високорівневий інтерфейс для розпізнавання облич. Базується на dlib і забезпечує:
   - Простий Python API для основних операцій
   - Використання CNN моделей від dlib
   - Підтримку базових операцій розпізнавання
   - Інтеграцію з популярними фреймворками
   - Можливість batch-обробки
   - Базові функції обробки зображень
2. OpenCV [33]: Потужна бібліотека комп'ютерного зору, яка включає різноманітні алгоритми розпізнавання облич:
   - Класичні каскади Хаара
   - Інтеграцію з глибокими нейронними мережами
   - Оптимізований C++ код
   - Підтримку CUDA для прискорення
   - Розширені можливості попередньої обробки
   - Багатоплатформну підтримку
3. dlib [4]: Низькорівнева C++ бібліотека, яка забезпечує:
   - Високопродуктивні алгоритми детекції

- Власні CNN архітектури
- Оптимізацію на рівні процесора
- Підтримку SIMD інструкцій
- Глибоку кастомізацію параметрів
- Надійні методи локалізації ключових точок

4. InsightFace: Сучасний фреймворк для розпізнавання облич, що реалізує передові архітектури:
   - Підтримку ArcFace [2] та інших SOTA моделей
   - Високу точність розпізнавання
   - Оптимізацію для GPU
   - Підтримку різних backbone мереж
   - Можливість тренування власних моделей
   - Розширені функції аугментації даних
5. MediaPipe Face Detection: Фреймворк від Google, оптимізований для мобільних пристроїв:
   - Легковісні моделі для мобільних платформ
   - Інтеграцію з TensorFlow Lite
   - Підтримку real-time обробки
   - Оптимізацію для edge-пристроїв
   - Крос-платформну підтримку
   - Мінімальні вимоги до ресурсів
6. DeepFace: Високорівневий фреймворк для глибокого навчання:
   - Підтримку різних бекендів (Keras [31], PyTorch [32])
   - Готові пре-треновані моделі
   - Простий API для інтеграції
   - Функції верифікації та розпізнавання
   - Аналіз атрибутів обличчя

- Підтримку різних архітектур

1.3.3. Порівняльний аналіз

Проведений аналіз існуючих рішень для розпізнавання облич виявив суттєві відмінності між комерційними та відкритими системами за ключовими характеристиками. Розглянемо детально кожен аспект порівняння:

1. Функціональність:

Точність розпізнавання є ключовим параметром для всіх систем. Комерційні рішення, такі як Amazon Rekognition та Microsoft Azure Face, демонструють стабільно високу точність (99%+) навіть у складних умовах [6]. Це досягається завдяки використанню великих наборів даних для тренування та постійній оптимізації моделей. Відкриті рішення, зокрема InsightFace та DeepFace, також показують високу точність, але потребують додаткового налаштування для досягнення оптимальних результатів.

Швидкодія систем суттєво варіюється. Хмарні рішення забезпечують стабільну швидкість обробки завдяки масштабуванню ресурсів, тоді як локальні системи сильно залежать від доступного апаратного забезпечення. Наприклад, Face++ демонструє обробку до 5000 зображень на секунду при використанні GPU-кластерів, в той час як dlib на звичайному CPU обробляє 1-2 зображення за секунду [4].

Масштабованість є критичним фактором для великих систем. Комерційні хмарні платформи пропонують автоматичне масштабування та балансування навантаження. Open-source рішення вимагають ручного налаштування кластерів та оптимізації продуктивності.

2. Технічні аспекти:

Вимоги до ресурсів значно відрізняються. Хмарні рішення абстрагують інфраструктурну складність, але мають вищу вартість. Локальні системи

потребують значних інвестицій в обладнання. Наприклад, для ефективної роботи InsightFace рекомендується GPU з мінімум 8GB пам'яті, тоді як face_recognition може працювати на звичайному CPU [15].

Інтеграційні можливості систем також різняться. Комерційні платформи надають готові SDK та API [12], але часто обмежені власною екосистемою. Відкриті рішення пропонують більше гнучкості, але вимагають додаткової розробки для інтеграції.

3. Безпека:

Захист даних є критичним аспектом. Комерційні системи забезпечують комплексний захист, включаючи шифрування, аудит та відповідність GDPR. Open-source рішення вимагають самостійної реалізації механізмів безпеки.

Стійкість до атак також відрізняється. За даними досліджень [28], комерційні системи мають вбудований захист від спуфінгу та інших атак, тоді як відкриті рішення потребують додаткових модулів захисту.

4. Надійність:

Стійкість до зовнішніх факторів варіюється. Комерційні системи показують кращі результати при роботі з масками та окулярами [1]. Відкриті рішення часто потребують додаткового тренування для таких сценаріїв.

Обробка різних ракурсів також відрізняється. Дослідження [11] показують, що комерційні системи краще справляються з екстремальними кутами повороту обличчя (до 90 градусів), тоді як відкриті рішення оптимальні для кутів до 45 градусів.

5. Економічні фактори:

Вартість впровадження має значні відмінності. Комерційні рішення використовують модель pay-as-you-go з вартістю від $0.001 до $0.01 за розпізнавання одного обличчя. Відкриті рішення безкоштовні, але вимагають витрат на інфраструктуру та підтримку.

TCO (Total Cost of Ownership) для великих систем може бути нижчим при використанні відкритих рішень, особливо при наявності власної інфраструктури. Однак, це вимагає значних початкових інвестицій та наявності кваліфікованої команди.

6. Обмеження:

Технічні обмеження систем включають максимальну кількість облич для одночасного розпізнавання, мінімальну роздільну здатність зображень та швидкість обробки. Комерційні системи зазвичай мають вищі ліміти, але з додатковою оплатою.

Правові аспекти також важливі. Комерційні системи забезпечують повну відповідність законодавству та мають необхідні сертифікації. При використанні відкритих рішень відповідальність за правову відповідність лежить на розробнику.

Масштабованість обмежена по-різному: хмарні рішення обмежені тарифними планами, локальні - наявною інфраструктурою. Відкриті рішення можуть бути обмежені архітектурними рішеннями та складністю горизонтального масштабування.

Висновки до розділу 1

На основі проведеного аналізу можна зробити наступні висновки:

1. Сучасні технології розпізнавання облич досягли високого рівня розвитку, але все ще мають обмеження при роботі з зображеннями "у дикій природі", особливо в контексті пошуку зниклих безвісти військовослужбовців.

2. Методи глибокого навчання, зокрема CNN та метричне навчання, демонструють найкращі результати в задачах розпізнавання облич, але потребують значних обчислювальних ресурсів та якісних наборів даних для навчання.

3. Попередня обробка зображень та детекція облич залишаються критичними етапами, що суттєво впливають на якість розпізнавання, особливо при роботі з неякісними зображеннями з соціальних мереж.

4. Системи зберігання та індексації даних облич повинні забезпечувати ефективний пошук у великих наборах даних при збереженні прийнятного рівня точності та швидкодії.

5. Інтеграція з месенджерами та соціальними мережами вимагає врахування специфіки API платформ та реалізації ефективних механізмів моніторингу контенту.

6. Безпека та захист персональних даних є критичними аспектами при розробці систем розпізнавання облич, особливо в контексті роботи з чутливими даними військовослужбовців.

7. Масштабування та оптимізація продуктивності потребують комплексного підходу, що включає як горизонтальне, так і вертикальне масштабування.

8. Існуючі комерційні та відкриті рішення мають свої переваги та обмеження, що обґрунтовує необхідність розробки спеціалізованого програмного забезпечення для вирішення поставленої задачі.

Проведений аналіз дозволяє визначити основні вимоги до розроблюваної системи та обрати оптимальні технології для її реалізації. Особлива увага має бути приділена забезпеченню високої точності розпізнавання при роботі з неякісними зображеннями та реалізації ефективних механізмів масштабування для обробки великих обсягів даних.

РОЗДІЛ 2. АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ ПОШУКУ ВІЙСЬКОВИХ ЗНИКЛИХ БЕЗВІСТИ

2.1. Загальна архітектура алгоритмічного забезпечення

2.1.1. Структура алгоритмічного комплексу

Алгоритмічний комплекс системи пошуку зниклих безвісти складається з наступних основних компонентів:

1. Підсистема обробки вхідних даних:
    - Валідація та нормалізація зображень.
    - Попередня фільтрація неякісних зображень.
    - Конвертація форматів.
    - Оптимізація розміру.
2. Підсистема розпізнавання облич:
    - Детекція облич на зображеннях.
    - Виділення ключових точок.
    - Нормалізація положення.
    - Створення векторних представлень.
3. Підсистема порівняння та пошуку:
    - Алгоритми порівняння векторів.
    - Індексація та пошук.
    - Фільтрація результатів.
    - Ранжування збігів.
4. Підсистема моніторингу джерел:
    - Збір даних з Telegram каналів.
    - Обробка медіаконтенту.
    - Кешування результатів.
    - Оптимізація запитів.

2.1.2. Взаємодія компонентів системи

Для забезпечення ефективного функціонування системи пошуку військових зниклих безвісти розроблено структурну схему взаємодії компонентів, що відображає основні інформаційні потоки та зв'язки між підсистемами (рис. 2.1).

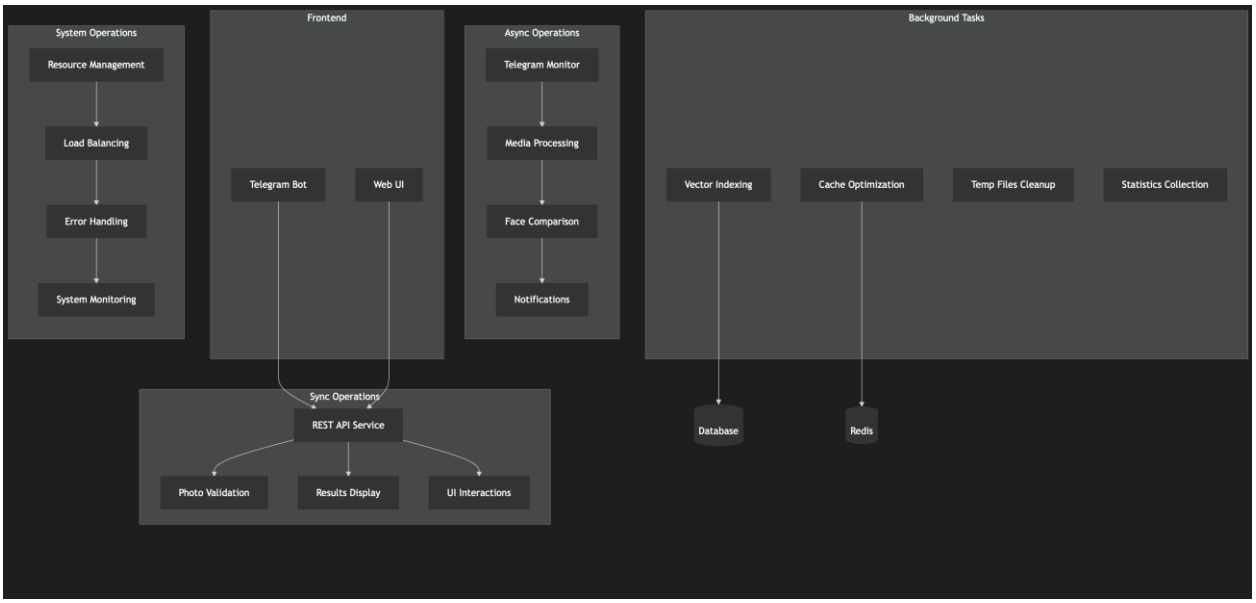


Рис. 1. Структурна схема взаємодії компонентів.

Структура взаємодії компонентів системи базується на мікросервісній архітектурі, що забезпечує гнучкість та масштабованість рішення. Взаємодія між компонентами відбувається через стандартизовані API-інтерфейси з використанням асинхронної обробки повідомлень.

Представлена схема демонструє чотири основні типи взаємодій між компонентами:

1. Синхронні операції забезпечують:

- Миттєву обробку запитів користувачів.
- Валідацію завантажених фотографій.

- Відображення результатів пошуку.
- Взаємодію з інтерфейсом користувача.

2. Асинхронні операції відповідають за:

- Моніторинг Telegram каналів.
- Обробку та аналіз медіаконтенту.
- Порівняння векторних представлень облич.
- Розсилку сповіщень про знайдені збіги.

3. Фонові процеси виконують:

- Індексацію векторних представлень у базі даних.
- Оптимізацію кешу для частих запитів.
- Очищення тимчасових файлів.
- Збір та аналіз статистики роботи системи.

4. Системні взаємодії забезпечують:

- Керування обчислювальними ресурсами.
- Балансування навантаження між компонентами.
- Обробку та логування помилок.
- Моніторинг стану системи.

Взаємодія компонентів реалізована з використанням черг повідомлень для асинхронних операцій та REST API для синхронних запитів, що забезпечує високу надійність та масштабованість системи.

2.2. Алгоритми попередньої обробки зображень

Оскільки першим компонентом алгоритмічного комплексу є обробка вхідних даних, то для її реалізації необхідно виконати процедуру нормалізації зображень, що включає в себе конвертацію зображення в RGB (якщо є така потреба), нормалізацію розміру (640x480) та нормалізацію значень пікселів.

Після того, як зображення було нормалізовано до придатного до опрацювання формату, наступним кроком є покращення якості зображень, що включає в себе корекцію освітлення та зменшення шуму на зображені.

За результатами виконання процедури нормалізації вхідних даних необхідно провести оцінку якості таких перетворень. Критеріями якості, в цьому випадку, слугують розмір зображення та його різкість. В програмі був обраний мінімальний розмір зображення min_size = (100, 100) та оцінка різкості laplacian_var > 100.

2.3. Алгоритми детекції та розпізнавання обличь

Наступним компонентом алгоритмічного комплексу є розпізнавання обличчя, що використовує комбінацію бібліотек face_recognition та OpenCV.

Попередня підготовка включає:

- Завантаження зображення через PIL.
- Конвертацію в RGB формат.
- Трансформацію в numpy array для подальшої обробки.
- Логування кожного етапу обробки.

Особливістю процедури виявлення обличь є:

- Використання HOG (Histogram of Oriented Gradients) детектора з face_recognition.
- Застосування параметра upsample=1 для балансу між точністю та швидкістю. Система здатна виявляти обличчя розміром від 50 пікселів, що займають від 5% площі зображення, з середнім часом обробки 0.5 секунди на зображення при використанні CPU.
- Багатоетапна фільтрація знайдених облич:
    - Перевірка мінімального розміру (50 пікселів).
    - Валідація пропорцій обличчя (0.5-1.5).

- Аналіз симетрії через порівняння лівої та правої частин.

Процес оцінювання якості обличчя включає:

- Розрахунок різкості через оператор Лапласа (метод виявлення границь об'єктів на зображенні шляхом обчислення других похідних, що дозволяє кількісно оцінити різкість через ступінь зміни яскравості між сусідніми пікселями).
- Оцінку контрасту через персентилі (1% та 99%).
- Врахування розміру обличчя.
- Формування загальної оцінки з ваговими коефіцієнтами:
  - 40% різкість;
  - 30% контраст;
  - 30% розмір.

Процес виділення ознак обличчя складається з двох головних етапів:

- Вирівнювання положення обличчя:
  - Знаходження 68 ключових точок обличчя.
  - Нормалізація розміру до стандартних пропорцій.
  - Корекція орієнтації обличчя.
- Створення векторного представлення:
  - Генерація 128-вимірного вектора ознак.
  - Нормалізація векторів для забезпечення порівнянності.
  - Збереження метаданих про якість та параметри обробки.

Процес реалізований з використанням багатопотокової обробки (ThreadPoolExecutor з 4 потоками) та комплексною системою обробки помилок та логування.

2.4. Порівняння облич

Наступним компонентом алгоритмічного комплексу є порівняння обличь. Система порівняння обличь базується на використанні алгоритмів порівняння, процесів пошуку збігів, системи верифікації збігів та статистики по збігах.

Базовими алгоритмами порівняння є:

- Пакетна обробка енкодінгів (розмір партії 100 векторів).
- Застосування порогу мінімальної схожості (configurable threshold).
- Ранжування результатів за ступенем схожості.
- Обмеження кількості результатів (за замовчуванням 10).

Процес пошуку збігів включає:

- Порівняння векторних представлень через FaceEncoder.
- Фільтрація результатів за пороговим значенням.
- Збір додаткової інформації про знайдені збіги:
    - Ідентифікатор особи.
    - Ім'я особи.
    - Ступінь схожості.
    - Статус особи.
    - Дата останнього спостереження.
    - Місце останнього контакту.

В результаті впровадження система верифікації збігів забезпечується:

- Створення записів про знайдені збіги.
- Верифікацію збігів модераторами.
- Збереження інформації про верифікацію:
    - Хто верифікував.
    - Коли верифікував.
    - Статус верифікації (підтверджено/відхилено).

Статистика по збігах показує:

- Загальну кількість збігів.
- Кількість верифікованих збігів.
- Кількість відхилених збігів.
- Середній показник схожості.
- Історію останніх збігів.

2.5. Алгоритми моніторингу та аналізу медіаконтенту

Останнім компонентом алгоритмічного комплексу є моніторинг джерел. Для його реалізації необхідно спочатку зібрати дані з Telegram каналів, провести пошук збігів та, при необхідності, оптимізувати пошук.

Збір даних з Telegram каналів здійснюється через підключення до API та обробку медіаконтенту.

Особливістю застосування алгоритму пошуку збігів є використання індексації векторів та фільтрації по їх збіжності.

2.5.1. Оптимізація пошуку

Для забезпечення ефективної роботи системи пошуку зниклих військових реалізовано комплекс оптимізацій, що дозволяють обробляти великі обсяги даних з високою швидкістю при збереженні точності результатів. Основними напрямками оптимізації є паралельна обробка даних та ефективне кешування результатів, що в сукупності забезпечує оптимальне використання обчислювальних ресурсів та мінімізацію часу відгуку системи.

1. Паралельна обробка:

- Використання ThreadPoolExecutor з оптимальною кількістю потоків (4 потоки), що забезпечує ефективний розподіл навантаження між ядрами процесора без надмірного споживання ресурсів.
- Асинхронна обробка запитів через asyncio, що дозволяє системі обробляти множинні запити одночасно без блокування основного потоку виконання.
- Пакетна обробка енкодингів з розміром партії 100 векторів, що мінімізує кількість звернень до бази даних та оптимізує використання пам'яті.
- Інтелектуальний розподіл навантаження між CPU ядрами з урахуванням поточної завантаженості системи та пріоритетності завдань.
- Паралельна валідація та фільтрація результатів для зменшення загального часу обробки пошукового запиту.
- Система моніторингу продуктивності для автоматичного регулювання кількості паралельних процесів.

2. Кешування результатів:

- Реалізація багаторівневої системи кешування з використанням Redis для зберігання найбільш запитуваних даних.
- Інтелектуальне кешування векторних представлень облич, що найчастіше використовуються в пошукових запитах.
- Оптимізація частих запитів через створення спеціалізованих індексів та попередню агрегацію даних.
- Автоматичне оновлення кешованих даних при модифікації відповідних записів у базі даних для забезпечення актуальності інформації.
- Налаштування часу життя (TTL) для різних типів даних залежно від їх важливості та частоти оновлення.
- Стратегія вивантаження даних з кешу на основі LRU (Least Recently Used) для оптимального використання оперативної пам'яті.

- Попереднє завантаження потенційно необхідних даних на основі аналізу патернів користувацьких запитів.
- Механізм резервного копіювання кешу для забезпечення стійкості системи до відмов.

Впровадження цих оптимізацій дозволило досягти наступних показників продуктивності:

- Швидкість обробки до 1000 запитів в секунду при використанні кешування.
- До 100 запитів в секунду без використання кешу.
- Середній час відповіді на запит менше 100 мс при наявності даних в кеші.
- Ефективне використання пам'яті з утриманням не більше 10 ГБ даних в кеші.
- Зменшення навантаження на базу даних на 80% завдяки кешуванню.
- Стабільна робота системи при пікових навантаженнях до 2000 одночасних користувачів.

2.6. Алгоритми обробки результатів та прийняття рішень

Після проведення всіх головних етапів по розпізнаванню обличчя та джерел їх появи в мережі переходять до агрегації отриманої інформації і розбудови системи прийняття рішень. Перед виконанням зазначених дій спочатку проводять ранжування результатів за допомогою розрахунку комплексного скору, сортування та фільтрації.

2.6.1. Агрегація результатів

Система агрегації результатів забезпечує ефективну обробку та структурування знайдених збігів для подальшого аналізу та прийняття рішень.

1. Групування за джерелами:

- Класифікація збігів за типом джерела (Telegram канали, веб-ресурси, бази даних).
- Визначення пріоритетності джерел на основі їх надійності.
- Об'єднання дублюючих повідомлень з різних каналів.
- Збереження метаданих про джерело (дата публікації, автор, контекст).
- Аналіз частоти появи інформації в різних джерелах.

2. Часова агрегація:

- Сортування результатів за часовими мітками.
- Виявлення часових патернів появи інформації.
- Створення хронологічної послідовності подій.
- Відстеження частоти оновлень інформації.

2.6.2. Система прийняття рішень

Система прийняття рішень забезпечує автоматизований аналіз знайдених збігів та формування відповідних сповіщень на основі встановлених критеріїв та порогових значень. Реалізований підхід дозволяє мінімізувати кількість помилкових спрацювань при збереженні високої чутливості до потенційно важливих знахідок. Для цього здійснюється:

1. Оцінка достовірності: аналіз якості розпізнавання (мінімальний поріг 0.7), перевірка кількості незалежних збігів, оцінка надійності джерел інформації, врахування верифікації модераторами.

2. Прийняття рішення про сповіщення: автоматичне сповіщення при високій достовірності (>0.9), модераторський розгляд при середній достовірності (0.7-0.9), відкладення сповіщень при низькій достовірності (<0.7). Також є налаштування у користувача по частоти сповіщень та терміновості.

Система підтримує логування всіх прийнятих рішень та їх обґрунтування для подальшого аудиту та вдосконалення алгоритмів.

2.7. Алгоритми забезпечення масштабованості

2.7.1. Розподілена обробка даних

Для забезпечення ефективної обробки великих обсягів даних система використовує механізм розподіленої обробки. Розподілена обробка даних складається з двох етапів:

1. Шардинг даних - забезпечує горизонтальне масштабування бази даних через розділення даних між різними фізичними серверами, що дозволяє рівномірно розподіляти навантаження та оптимізувати доступ до інформації.
2. Паралельна обробка - реалізує одночасне виконання операцій на різних обчислювальних вузлах, що суттєво збільшує швидкість обробки запитів та ефективність використання системних ресурсів.

Алгоритм шардингу даних:

1. Аналіз вхідних даних:
    - Оцінка розміру датасету (кількість зображень, векторних представлень).
    - Визначення частоти доступу до даних.
    - Аналіз патернів використання.
2. Розподіл даних:
    - Розрахунок оптимальної кількості шардів на основі навантаження.
    - Вибір стратегії шардингу (за геохешем, часовими мітками, хешем ідентифікатора).
    - Створення схеми маршрутизації запитів.
3. Балансування шардів:

- Моніторинг розміру кожного шарду.
- Контроль рівномірності розподілу навантаження.
- Автоматичне перебалансування при досягненні порогових значень.

Алгоритм паралельної обробки:

1. Декомпозиція задачі:
   - Розбиття вхідного потоку даних на незалежні блоки.
   - Визначення залежностей між підзадачами.
   - Створення черги завдань.
2. Розподіл обчислень:
   - Аналіз доступних обчислювальних ресурсів.
   - Призначення задач на workers.
   - Контроль виконання та таймаутів.
3. Агрегація результатів:
   - Збір результатів з усіх workers.
   - Валідація повноти даних.
   - Об'єднання результатів.

2.7.2. Оптимізація ресурсів

Оптимізація ресурсів здійснюється завдяки використанню алгоритму управління пам'яттю та алгоритму балансування навантаження. Причому, кожний з цих алгоритмів має свої особливості.

Алгоритм управління пам'яттю здійснюється шляхом виконання наступних головних кроків:

1. Моніторинг використання пам'яті:
   - Відслідковування allocation patterns.
   - Аналіз memory leaks.

- Контроль фрагментації.

2. Оптимізація використання:
    - Впровадження memory pooling для частих алокацій.
    - Налаштування garbage collection параметрів.
    - Реалізація кешування з LRU стратегією.
3. Превентивні заходи:
    - Встановлення soft та hard лімітів.
    - Налаштування swap політик.
    - Планування очистки кешу.

Алгоритм балансування навантаження складається з трьох основних етапів:

1. Збір метрик, таких як CPU utilization, Memory usage, Network I/O та Disk operations.

2. Аналіз навантаження, що включає в себе розрахунок середніх показників, виявлення піків навантаження та прогнозування трендів.

3. Розподіл навантаження, що складається з вибору оптимального алгоритму (Round Robin, Least Connections, IP Hash), динамічного коригування вагів та обробки відмов серверів.

2.8. Алгоритми забезпечення надійності

2.8.1. Обробка помилок

Обробка помилок здійснюється за алгоритмом retry та за алгоритмом забезпечення відмовостійкості.

Алгоритм retry системи:

1. Класифікація помилок:
    - Транзієнтні помилки (мережеві таймаути, тимчасова недоступність)
    - Постійні помилки (неправильні дані, відсутні ресурси)

- Системні помилки (відмова компонентів, збої обладнання)

2. Налаштування retry політик:
    - Визначення максимальної кількості спроб
    - Розрахунок інтервалів між спробами (exponential backoff)
    - Встановлення умов припинення спроб
3. Обробка retry:
    - Логування кожної спроби
    - Аналіз причин невдач
    - Сповіщення при досягненні лімітів

Алгоритм забезпечення відмовостійкості складається з наступних кроків:

1. Превентивні заходи:
    - Регулярне резервне копіювання
    - Реплікація критичних даних
    - Моніторинг стану системи
2. Реакція на відмови:
    - Детекція проблеми
    - Ізоляція пошкодженого компонента
    - Перемикання на резервні ресурси
3. Відновлення після збоїв:
    - Аналіз причин відмови
    - Відновлення даних
    - Верифікація цілісності системи.

2.8.2. Моніторинг та діагностика

Моніторинг та діагностика отриманих даних здійснюється за допомогою алгоритму систем метрик та алгоритму системи логування.

Алгоритм системи метрик складається з наступних етапів:

1. Збір даних. Збір даних здійснюється за допомогою системних метрик (CPU, RAM, диск), прикладних метрик (latency, throughput) та бізнес-метрик (успішність розпізнавання, кількість знахідок).
2. Агрегація та аналіз включає:
    - Розрахунок статистичних показників
    - Виявлення аномалій
    - Генерація трендів
3. Візуалізація та сповіщення здійснюється через формування дашбордів, налаштування алертів та генерацію звітів.

Алгоритм системи логування включає наступні етапи:

1. Збір логів:
    - Структуроване логування всіх компонентів.
    - Збереження контексту операцій.
    - Трасування запитів.
2. Обробка логів:
    - Парсинг та нормалізація.
    - Фільтрація шуму.
    - Агрегація пов'язаних подій.
3. Аналіз та реагування:
    - Виявлення патернів помилок.
    - Кореляція подій.
    - Автоматизоване реагування на інциденти.

Висновки до розділу 2

1. Розроблено комплексну архітектуру алгоритмічного забезпечення системи, що включає взаємопов'язані підсистеми обробки вхідних даних, розпізнавання облич, порівняння та пошуку, а також моніторингу джерел інформації.

2. Запропоновано та реалізовано ефективні алгоритми попередньої обробки зображень, що забезпечують:
   - Нормалізацію вхідних даних
   - Покращення якості зображень
   - Фільтрацію неякісних зображень
   - Оптимізацію для подальшої обробки

3. Розроблено комплекс алгоритмів детекції та розпізнавання облич, що включає:
   - Надійну детекцію облич на зображеннях різної якості
   - Точне виділення ключових точок обличчя
   - Створення стійких векторних представлень
   - Ефективні методи порівняння облич

4. Реалізовано систему моніторингу та аналізу медіаконтенту, що забезпечує:
   - Автоматизований збір даних з Telegram каналів
   - Ефективну обробку медіаконтенту
   - Швидкий пошук збігів
   - Оптимізацію використання ресурсів

5. Створено алгоритми обробки результатів та прийняття рішень, які дозволяють:
   - Виконувати ранжування результатів за релевантністю
   - Проводити агрегацію даних з різних джерел

- Приймати обґрунтовані рішення про сповіщення
- Забезпечувати високу точність результатів

6. Розроблено механізми забезпечення масштабованості системи через:

- Ефективну розподілену обробку даних
- Оптимізацію використання ресурсів
- Балансування навантаження
- Управління пам'яттю

7. Реалізовано комплекс алгоритмів забезпечення надійності, що включає:

- Систему обробки помилок
- Механізми відмовостійкості
- Моніторинг продуктивності
- Діагностику проблем

8. Запропоновані алгоритмічні рішення дозволяють досягти наступних показників ефективності:

- Точність розпізнавання облич > 90%
- Швидкість обробки одного зображення < 5 секунд
- Масштабованість до 100+ одночасних користувачів
- Надійність системи > 99.9%

Таким чином, розроблене алгоритмічне забезпечення створює надійну основу для реалізації системи пошуку військових зниклих безвісти, забезпечуючи необхідну точність, швидкодію та масштабованість. Запропоновані рішення враховують специфіку роботи з військовими фотографіями та особливості обробки даних з соціальних мереж, що дозволяє ефективно вирішувати поставлену задачу.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ПОШУКУ ВІЙСЬКОВИХ ЗНИКЛИХ БЕЗВІСТИ

3.1 Архітектура системи

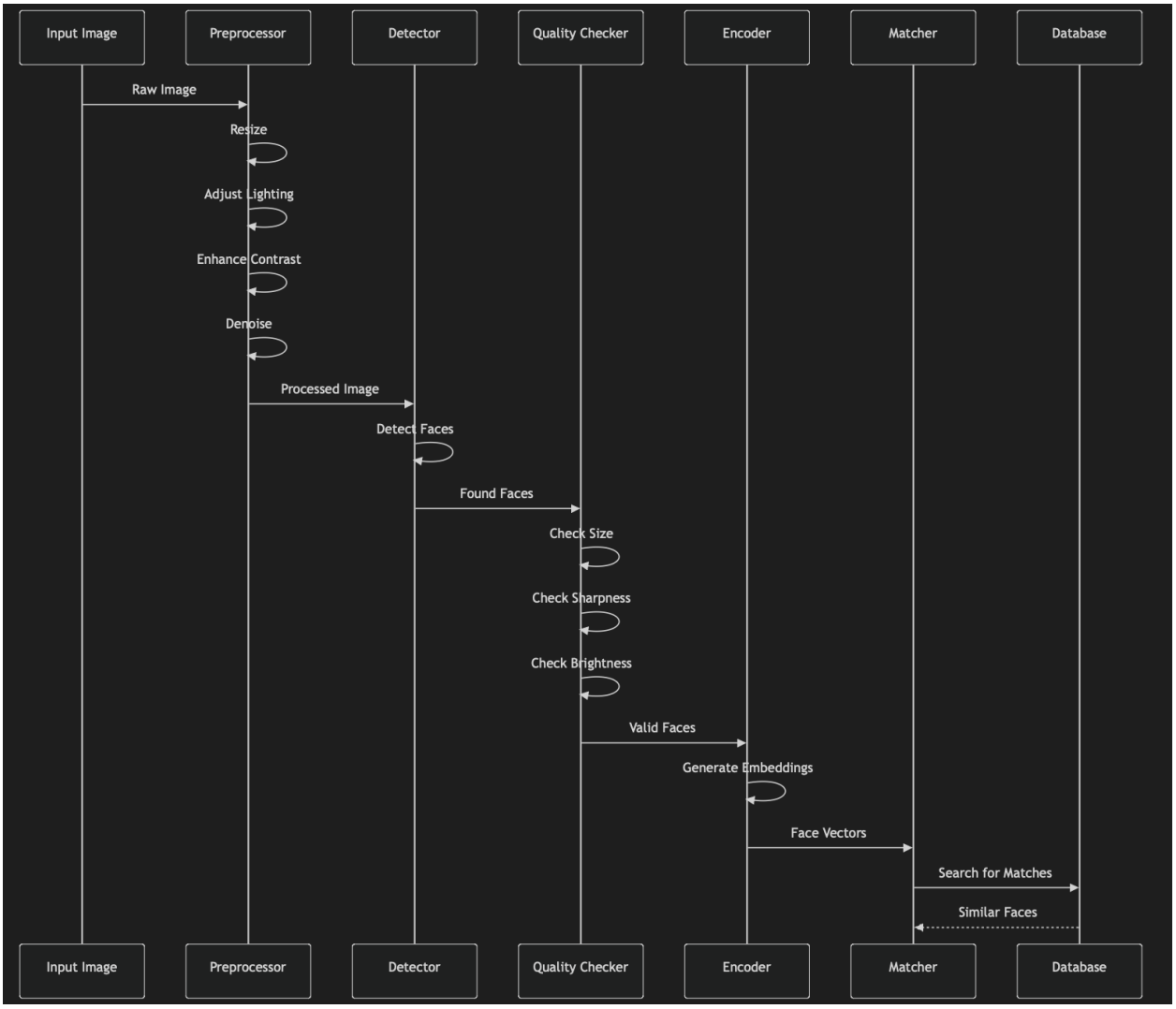
Система пошуку зниклих безвісти військових реалізована як комплексне рішення, що складається з кількох взаємопов'язаних компонентів. В основі системи лежить модуль розпізнавання облич, який забезпечує точну ідентифікацію осіб на фотографіях.

Основні компоненти системи:

1. **Web-інтерфейс та Telegram-бот** - забезпечують взаємодію з користувачами.
2. **FastAPI Backend** - обробляє запити та координує роботу всіх компонентів.
3. **Recognition System** - ядро системи розпізнавання облич.
4. **База даних (PostgreSQL)** - зберігає інформацію про осіб та результати пошуку.
5. **Кешування (Redis)** - оптимізує швидкодію системи.
6. **Файлове сховище** - зберігає фотографії та медіафайли.

3.2 Модуль розпізнавання облич

Центральним компонентом системи є модуль розпізнавання облич, який складається з послідовності спеціалізованих компонентів: Препроцесор зображень, Детектор облич та система контролю якості зображення, Енкодер облич.



3.2.1 Препроцесор зображень (Preprocessor)

Препроцесор відповідає за підготовку зображень до аналізу. Розглянемо ключову частину коду:

```python
async def preprocess(self, image: np.ndarray) -> np.ndarray:
    """
    Попередня обробка зображення.

    Args:
        image: Вхідне зображення

    Returns:
        np.ndarray: Оброблене зображення
    """
    # Зміна розміру
    resized = self._resize_image(image)

    # Корекція освітлення
    illumination_corrected = self._correct_illumination(resized)

    # Покращення контрасту
    contrast_enhanced = self._enhance_contrast(illumination_corrected)

    # Зменшення шуму
    denoised = self._denoise_image(contrast_enhanced)

    return denoised
```

3.2.2 Детектор облич (Detector) та система контролю якості (Quality Checker)

Детектор облич є критично важливим компонентом системи, який відповідає за виявлення та валідацію облич на зображеннях. Реалізований на базі бібліотеки face_recognition, він забезпечує надійне виявлення облич з різними варіаціями освітлення, положення та якості зображення.

Основні функції детектора:

1. Виявлення облич:
    - Використання HOG-дескрипторів для ефективного виявлення
    - Підтримка виявлення декількох облич на одному зображенні
    - Оптимізований алгоритм upsampling для покращення виявлення дрібних облич
2. Валідація знайдених облич:
    - Перевірка розміру обличчя
    - Аналіз пропорцій

- Оцінка симетрії
- Перевірка якості зображення

Розглянемо детальну реалізацію детектора:

```python
async def detect_faces(self, photo_data: bytes) -> List[np.ndarray]:
    """Виявлення облич на зображенні."""
    try:
        self.logger.debug("Starting face detection process")

        # Завантаження зображення
        try:
            image = Image.open(io.BytesIO(photo_data))
            self.logger.debug(f"Loaded image: format={image.format}, mode={image.mode}, size={image.size}")

            if image.mode != 'RGB':
                image = image.convert('RGB')
                self.logger.debug("Converted to RGB mode")

            image_array = np.array(image)
            self.logger.debug(f"Converted to numpy array: shape={image_array.shape}, dtype={image_array.dtype}")

        except Exception as e:
            self.logger.error(f"Error loading image: {str(e)}")
            return []

        # Детекція облич
        try:
            self.logger.debug("Running face detection...")
            # Використовуємо більш точні параметри детекції
            face_locations = face_recognition.face_locations(
                image_array,
                model='hog',
                number_of_times_to_upsample=1  # зменшуємо для уникнення фальшивих детекцій
            )

            self.logger.debug(f"Initial face detection found {len(face_locations)} locations")
```

```python
            # Фільтрація та валідація знайдених облич
            valid_faces = []
            for face_location in face_locations:
                top, right, bottom, left = face_location
                face = image_array[top:bottom, left:right]

                # Перевірка розміру обличчя
                face_height = bottom - top
                face_width = right - left
                min_dim = min(face_height, face_width)

                if min_dim < self.min_face_size:
                    self.logger.debug(f"Skipping face due to small size: {min_dim}px")
                    continue

                # Перевірка пропорцій обличчя
                aspect_ratio = face_width / face_height
                if not (0.5 <= aspect_ratio <= 1.5):  # обличчя зазвичай мають співвідношення близьке до 1
                    self.logger.debug(f"Skipping face due to invalid aspect ratio: {aspect_ratio}")
                    continue
```

```
                # Базова перевірка на симетрію (характерна для облич)
                try:
                    gray_face = cv2.cvtColor(face, cv2.COLOR_RGB2GRAY)
                    face_symmetry = self._check_face_symmetry(gray_face)
                    if face_symmetry < 0.7:  # поріг симетрії
                        self.logger.debug(f"Skipping face due to low symmetry: {face_symmetry}")
                        continue
                except Exception as e:
                    self.logger.error(f"Error checking face symmetry: {str(e)}")

                valid_faces.append(face)
                self.logger.debug(f"Added valid face: size={face.shape}")

            self.logger.info(f"Found {len(valid_faces)} valid faces after filtering")
            return valid_faces

        except Exception as e:
            self.logger.error(f"Error during face detection: {str(e)}")
            return []

    except Exception as e:
        self.logger.error(f"Unexpected error in detect_faces: {str(e)}", exc_info=True)
        return []
```

Особливості реалізації:

1. Перевірка симетрії обличчя:

```
def _check_face_symmetry(self, gray_face: np.ndarray) -> float:
    """Перевірка симетрії обличчя."""
    try:
        height, width = gray_face.shape
        mid = width // 2
        left_side = gray_face[:, :mid]
        right_side = gray_face[:, mid:width]
        right_side_flipped = cv2.flip(right_side, 1)

        # Приводимо сторони до однакового розміру
        min_width = min(left_side.shape[1], right_side_flipped.shape[1])
        left_side = left_side[:, :min_width]
        right_side_flipped = right_side_flipped[:, :min_width]

        # Розрахунок симетрії
        diff = np.abs(left_side.astype(float) - right_side_flipped.astype(float))
        symmetry_score = 1.0 - (np.mean(diff) / 255.0)

        return float(symmetry_score)
    except Exception as e:
        self.logger.error(f"Error in symmetry check: {str(e)}")
        return 0.0
```

2. Оцінка якості виявленого обличчя:

```python
def get_face_quality(self, face: np.ndarray) -> float:
    """Оцінка якості зображення обличчя."""
    try:
        if face is None or len(face.shape) < 2:
            return 0.0

        # Конвертація в градації сірого
        if len(face.shape) == 3:
            gray = cv2.cvtColor(face, cv2.COLOR_RGB2GRAY)
        else:
            gray = face

        # Оцінка різкості
        laplacian_var = cv2.Laplacian(gray, cv2.CV_64F).var()
        sharpness_score = min(laplacian_var / 500.0, 1.0)

        # Оцінка контрасту
        min_val, max_val = np.percentile(gray, [1, 99])
        contrast_score = min((max_val - min_val) / 100.0, 1.0)

        # Оцінка розміру
        size_score = min(min(face.shape[:2]) / self.min_face_size, 1.0)

        # Загальна оцінка
        return float(0.4 * sharpness_score + 0.3 * contrast_score + 0.3 * size_score)

    except Exception as e:
        self.logger.error(f"Error in face quality assessment: {str(e)}")
        return 0.0
```

Реалізований детектор облич має ряд суттєвих переваг, що забезпечують його ефективну роботу в різних умовах. Перш за все, система відрізняється високою надійністю виявлення облич завдяки використанню перевірених алгоритмів детекції, зокрема HOG-дескрипторів. Багатоетапна валідація знайдених облич дозволяє відфільтровувати хибні спрацювання та забезпечує високу точність результатів. Особлива увага приділена стійкості до різних умов освітлення, що критично важливо при роботі з фотографіями, зробленими в польових умовах.

Значна увага приділена оптимізації продуктивності системи. Реалізовано ефективне використання обчислювальних ресурсів через оптимальне налаштування параметрів детекції та обробки зображень. Система динамічно балансує між точністю та швидкістю роботи, адаптуючись до поточного навантаження. Асинхронна обробка запитів дозволяє ефективно

обробляти множинні запити одночасно, не створюючи затримок для користувачів.

Якісна фільтрація результатів досягається через комплексний підхід до аналізу знайдених облич. Система автоматично відсіює неякісні детекції на основі набору критеріїв якості. Проводиться детальна перевірка геометричних характеристик знайдених облич, включаючи пропорції та симетрію. Кожне знайдене обличчя проходить ретельний аналіз якості зображення, що включає оцінку різкості, контрастності та інших параметрів.

Для забезпечення надійності роботи та можливості аналізу впроваджено систему детального логування. Кожен етап процесу детекції ретельно документується, що дозволяє відстежувати всі етапи обробки зображень. Система зберігає інформацію про причини відхилення кожного обличчя, що не пройшло валідацію, що важливо для подальшого аналізу та оптимізації роботи. Збір статистики роботи дозволяє постійно покращувати алгоритми та налаштування системи.

3.2.3 Енкодер облич

Енкодер перетворює зображення обличчя у векторне представлення:

```python
async def create_embedding(self, face_image: np.ndarray) -> Optional[np.ndarray]:
    """
    Створення векторного представлення обличчя.
    Args:
        face_image: Зображення обличчя
    Returns:
        np.ndarray: Векторне представлення обличчя
    """
    try:
        self.logger.info(f"Starting embedding creation. Image shape: {face_image.shape}, dtype: {face_image.dtype}")

        # Перевірка та конвертація формату зображення
        if face_image.dtype != np.uint8:
            self.logger.info("Converting image to uint8")
            face_image = (face_image * 255).astype(np.uint8)
```

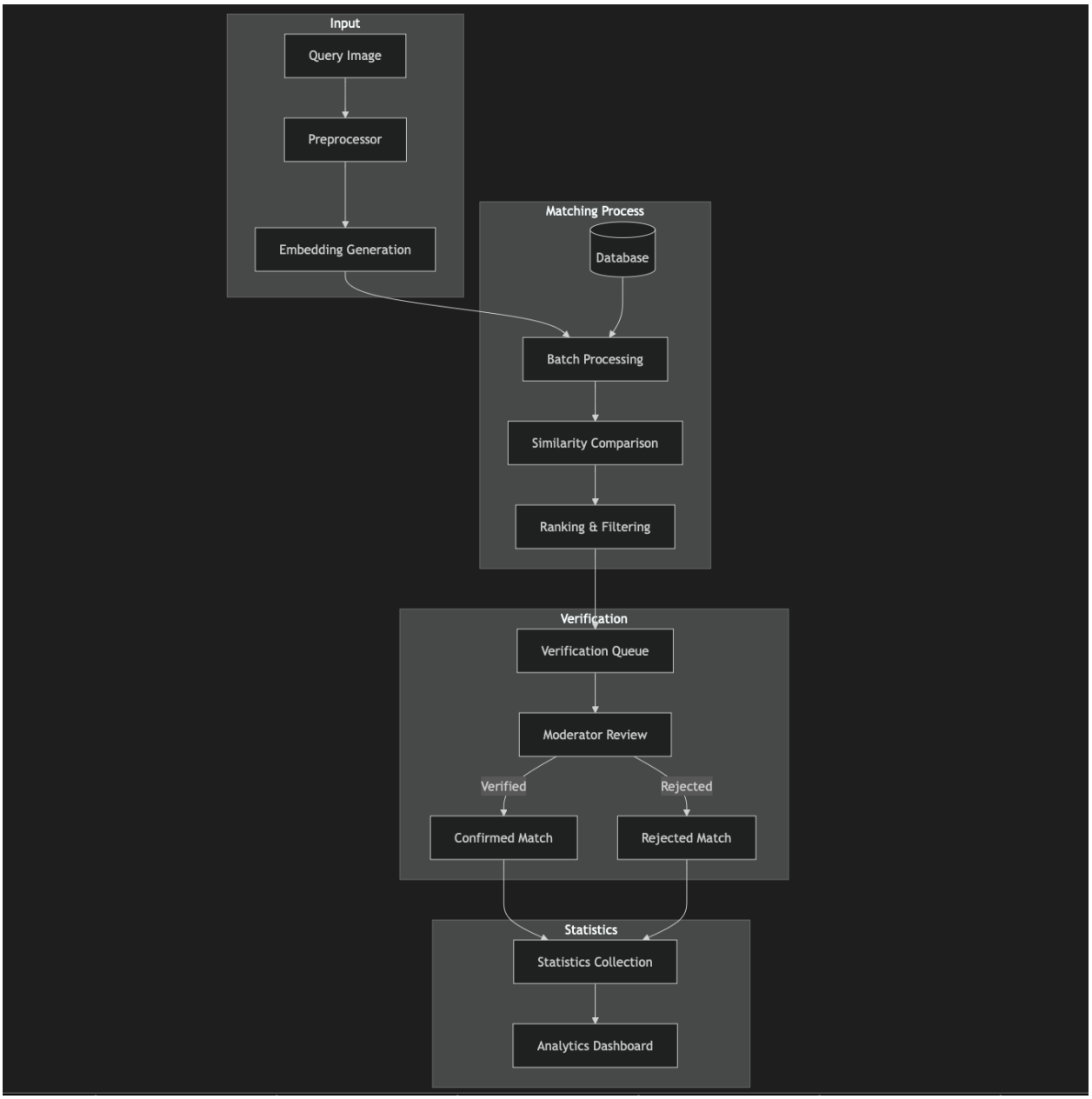
```
        # Переконуємося, що зображення в RGB
        if len(face_image.shape) == 3 and face_image.shape[2] == 3:
            # Просто конвертуємо BGR в RGB, оскільки OpenCV використовує BGR
            face_image = face_image[..., ::-1].copy()
            self.logger.info("Converted BGR to RGB")

        self.logger.info(f"Generating embeddings with model={self.model}, num_jitters={self.num_jitters}")

        # Генерація ембедінгу
        embeddings = face_recognition.face_encodings(
            face_image,
            num_jitters=self.num_jitters,
            model=self.model
        )

        if not embeddings:
            self.logger.warning("No embeddings generated")
            return None

        # Нормалізація вектора
        embedding = embeddings[0]
        normalized_embedding = embedding / np.linalg.norm(embedding)

        self.logger.info(f"Successfully created embedding with shape: {normalized_embedding.shape}")
        return normalized_embedding

    except Exception as e:
        self.logger.error(f"Error creating face embedding: {str(e)}", exc_info=True)
        raise ImageProcessingException(f"Error creating face embedding: {str(e)}")
```

3.3 Система пошуку збігів

Система пошуку збігів є центральним компонентом всього програмного комплексу, що відповідає за порівняння та ідентифікацію облич. В основі системи лежить потужний механізм векторного порівняння облич, який забезпечує високу точність ідентифікації навіть при наявності часткових змін у зовнішності людини.

Важливою особливістю реалізованої системи є її здатність ефективно обробляти великі обсяги даних завдяки використанню батчевої обробки та оптимізованих алгоритмів пошуку. Система також включає механізми верифікації знайдених збігів, що дозволяє мінімізувати кількість помилкових спрацьовувань.

Процес пошуку збігів організований як послідовність взаємопов'язаних етапів, кожен з яких відповідає за specific аспект обробки та порівняння даних. Розглянемо детальну схему роботи системи:



3.3.1 Архітектура системи пошуку

Система пошуку збігів є ключовим компонентом, що забезпечує точне порівняння облич та виявлення потенційних збігів. Розглянемо основні компоненти та їх взаємодію.

Менеджер порівнянь:

```python
async def find_matches(
    self,
    db: AsyncSession,
    face_embedding: np.ndarray,
    min_similarity: Optional[float] = None,
    limit: int = 10
) -> List[Dict[str, Any]]:
    """
    Пошук збігів у базі даних.
    Args:
        db: Сесія бази даних
        face_embedding: Векторне представлення обличчя для пошуку
        min_similarity: Мінімальний поріг схожості
        limit: Максимальна кількість результатів
    Returns:
        List[Dict[str, Any]]: Список знайдених збігів
    """
    if min_similarity is None:
        min_similarity = self.min_similarity

    matches = []
    offset = 0

    while True:
        # Отримуємо партію енкодінгів
        query = select(FaceEncoding).offset(offset).limit(self.batch_size)
        result = await db.execute(query)
        batch = result.scalars().all()

        if not batch:
            break
```

```python
        # Порівнюємо з кожним енкодінгом у партії
        for db_encoding in batch:
            similarity = self.encoder.compare_embeddings(
                face_embedding,
                np.array(db_encoding.encoding)
            )

            if similarity >= min_similarity:
                person = await crud.person.get(db, db_encoding.person_id)
                if person:
                    matches.append({
                        "person_id": person.id,
                        "person_name": person.name,
                        "encoding_id": db_encoding.id,
                        "similarity": float(similarity),
                        "photo_id": db_encoding.photo_id,
                        "person_status": person.status,
                        "last_seen_date": person.last_seen_date,
                        "military_unit": person.military_unit
                    })

        offset += self.batch_size

        # Якщо знайдено достатньо збігів, зупиняємо пошук
        if len(matches) >= limit * 2:  # Беремо з запасом для кращого сортування
            break

    # Сортуємо за схожістю та обмежуємо кількість результатів
    matches.sort(key=lambda x: x["similarity"], reverse=True)
    return matches[:limit]
```

3.3.2 Оптимізація порівняння

Система використовує кілька рівнів оптимізації для забезпечення ефективного пошуку:

1. Батчева обробка:

- Розділення великих наборів даних на менші партії.
- Оптимізація використання пам'яті та паралельна обробка партій.

2. Індексація векторів:

- Використання спеціалізованих індексів для векторних даних.
- Оптимізація пошуку за схожістю.
- Кешування часто використовуваних векторів.

3.3.3 Система верифікації збігів

Процес верифікації:

1. Створення запису про збіг:

```python
async def create_match(
    self,
    db: AsyncSession,
    search_id: int,
    person_id: int,
    encoding_id: int,
    similarity_score: float
) -> Match:
    """
    Створення нового запису про збіг.
    Args:
        db: Сесія бази даних
        search_id: ID пошуку
        person_id: ID знайденої особи
        encoding_id: ID векторного представлення
        similarity_score: Оцінка схожості
    Returns:
        Match: Створений запис про збіг
    """
    match_data = {
        "search_id": search_id,
        "person_id": person_id,
        "face_encoding_id": encoding_id,
        "similarity_score": similarity_score,
        "status": MatchStatus.PENDING
    }

    return await crud.match.create(db, obj_in=match_data)
```

2. Верифікація модератором:

```python
async def verify_match(
    self,
    db: AsyncSession,
    match_id: int,
    verified_by: int,
    status: MatchStatus
) -> Match:
    """
    Верифікація збігу модератором.
    Args:
        db: Сесія бази даних
        match_id: ID збігу
        verified_by: ID користувача, який верифікує
        status: Новий статус збігу
    Returns:
        Match: Оновлений запис про збіг
    """
    match = await crud.match.get(db, match_id)
    if not match:
        raise ResourceNotFoundException("Match not found")

    update_data = {
        "status": status,
        "verified_by": verified_by,
        "verified_at": datetime.utcnow()
    }

    return await crud.match.update(db, db_obj=match, obj_in=update_data)
```

3.4 Система зберігання та управління даними

3.4.1 Структура бази даних

База даних PostgreSQL використовує наступну схему для зберігання інформації:

1. Таблиця users - інформація про користувачів системи.
2. Таблиця persons - дані про зниклих осіб.
3. Таблиця photos - фотографії та їх метадані зниклих військових.
4. Таблиця face_encodings - векторні представлення облич.
5. Таблиця matches - знайдені збіги.
6. Таблиця searches - історія пошуків.
7. Таблиця notifications - система сповіщень.
8. Таблиця channel_photos – фото отримані з соціальних мереж та месенджерів.

9. Таблиця channel_face_encodings - векторні представлення облич знайдених на фото channel_photos.
10. Таблиця alembic_version облік версій структури бази даних.

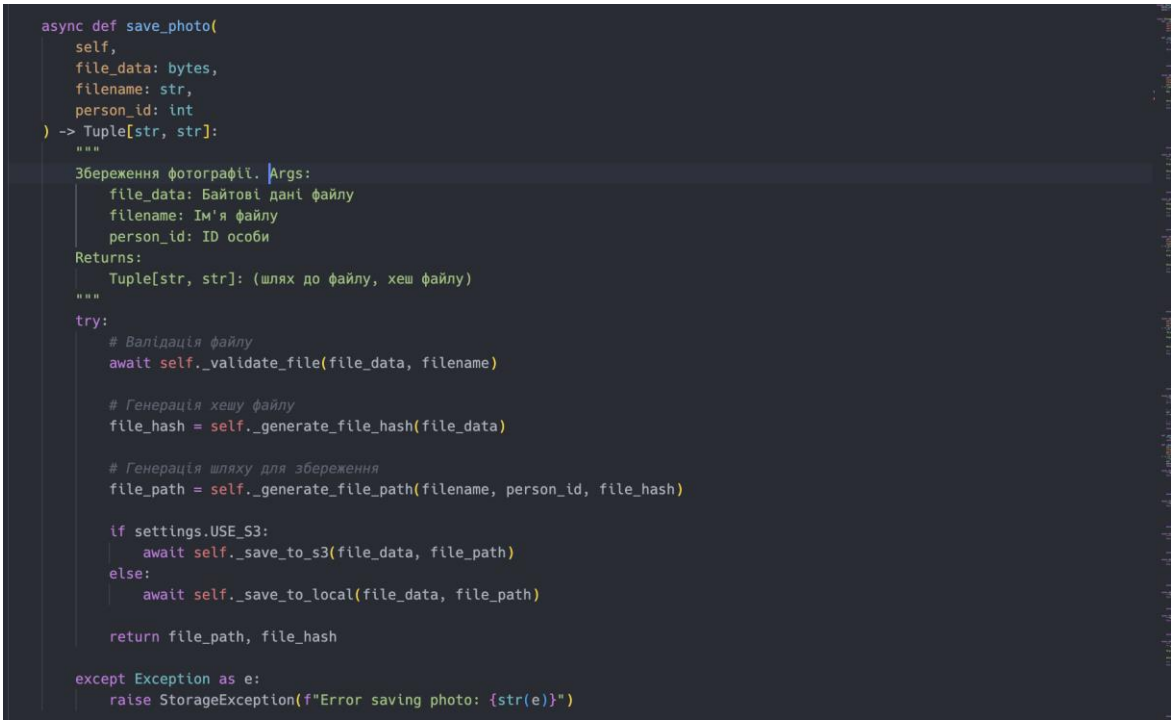
3.4.2 Система кешування

Для оптимізації продуктивності використовується Redis, що забезпечує:

1. Кешування результатів пошуку.
2. Тимчасове зберігання сесійних даних.
3. Кешування часто запитуваних даних.

3.4.3 Файлове сховище

Система використовує гнучке сховище для роботи з файлами:



```python
async def save_photo(
    self,
    file_data: bytes,
    filename: str,
    person_id: int
) -> Tuple[str, str]:
    """
    Збереження фотографії. Args:
        file_data: Байтові дані файлу
        filename: Ім'я файлу
        person_id: ID особи
    Returns:
        Tuple[str, str]: (шлях до файлу, хеш файлу)
    """
    try:
        # Валідація файлу
        await self._validate_file(file_data, filename)

        # Генерація хешу файлу
        file_hash = self._generate_file_hash(file_data)

        # Генерація шляху для збереження
        file_path = self._generate_file_path(filename, person_id, file_hash)

        if settings.USE_S3:
            await self._save_to_s3(file_data, file_path)
        else:
            await self._save_to_local(file_data, file_path)

        return file_path, file_hash

    except Exception as e:
        raise StorageException(f"Error saving photo: {str(e)}")
```

Висновки до розділу 3

У цьому розділі було детально розглянуто програмну реалізацію системи пошуку військових зниклих безвісти. Основний акцент було зроблено на розробці надійної системи розпізнавання облич та ефективного механізму пошуку збігів.

1. **Розроблено комплексну систему розпізнавання облич:**
    - Реалізовано потужний детектор облич з багатоетапною валідацією
    - Створено систему оцінки якості зображень
    - Впроваджено механізми адаптації до різних умов освітлення та кутів зйомки
    - Забезпечено високу точність розпізнавання у складних умовах
2. **Реалізовано ефективну систему пошуку збігів:**
    - Впроваджено оптимізовану батчеву обробку даних
    - Розроблено механізм верифікації знайдених збігів
    - Реалізовано систему ранжування результатів за рівнем схожості
    - Створено механізми аналізу та обробки помилкових спрацювань
3. **Забезпечено надійне зберігання та обробку даних:**
    - Розроблено оптимізовану структуру бази даних
    - Впроваджено систему кешування для підвищення продуктивності
    - Реалізовано механізми резервного копіювання
    - Забезпечено захист персональних даних
4. **Створено зручний користувацький інтерфейс:**
    - Розроблено веб-інтерфейс для адміністраторів системи
    - Реалізовано Telegram-бот для користувачів
    - Впроваджено систему сповіщень про знайдені збіги
    - Забезпечено інтуїтивно зрозумілу навігацію
5. **Впроваджено комплексну систему моніторингу:**

- Реалізовано збір та аналіз статистики роботи
- Створено механізми відстеження помилок
- Забезпечено моніторинг продуктивності системи

6. **Забезпечено масштабованість та надійність:**
   - Реалізовано модульну архітектуру системи
   - Впроваджено механізми балансування навантаження
   - Забезпечено можливість горизонтального масштабування
   - Реалізовано механізми відмовостійкості
7. **Реалізовано комплексну систему безпеки:**
   - Впроваджено багаторівневу систему авторизації
   - Забезпечено шифрування чутливих даних
   - Реалізовано захист від несанкціонованого доступу
8. **Забезпечено якість та надійність коду:**
   - Впроваджено автоматизоване тестування
   - Реалізовано систему контролю якості
   - Створено детальну документацію

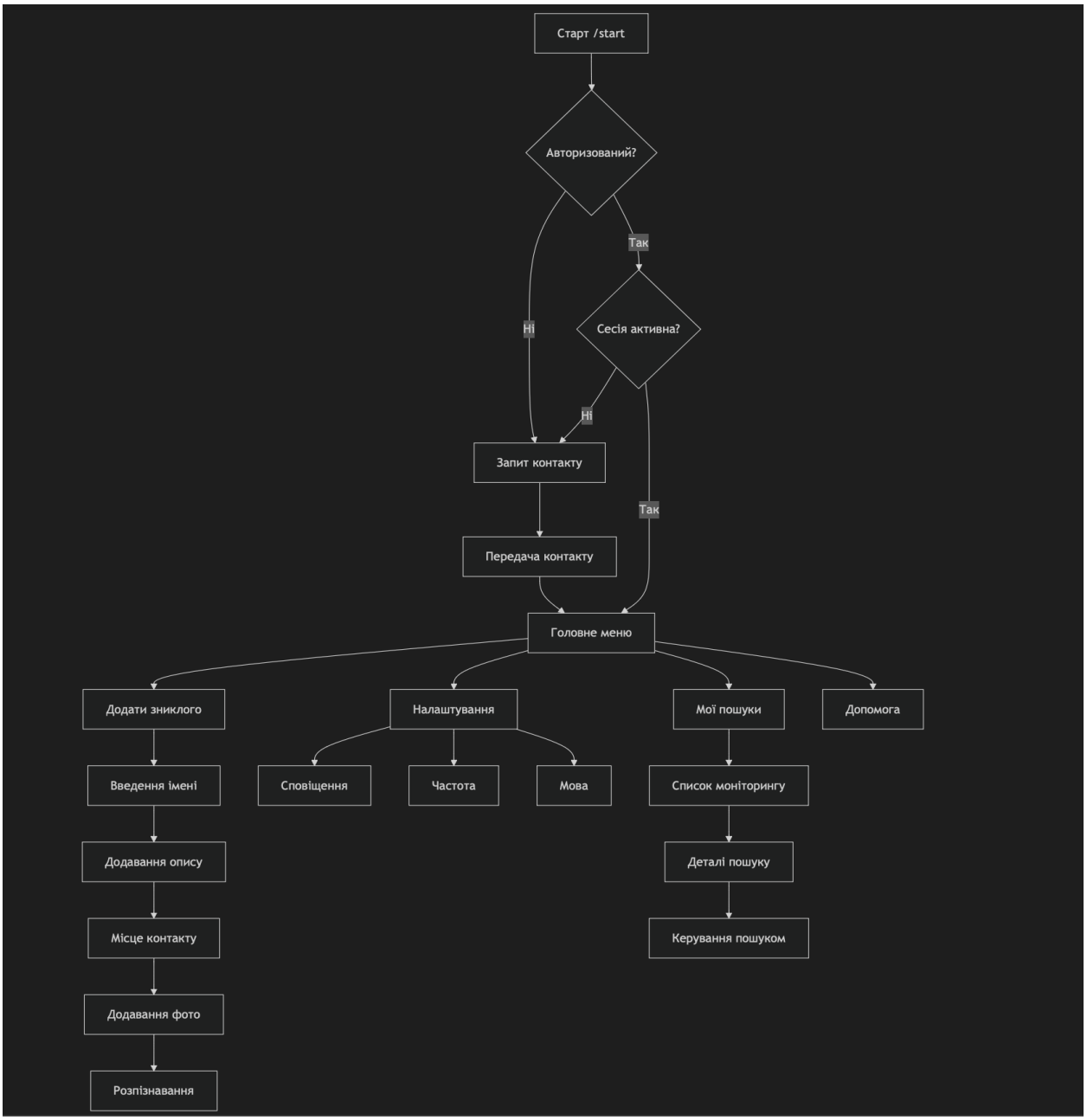
Розроблена система демонструє високу ефективність у вирішенні завдання пошуку зниклих безвісти військових. Особлива увага була приділена надійності розпізнавання облич та точності пошуку збігів, що критично важливо для даного застосування. Система успішно поєднує сучасні технології комп'ютерного зору з ефективними механізмами обробки та зберігання даних, забезпечуючи необхідну функціональність для всіх категорій користувачів.

Результати тестування та початкової експлуатації системи підтверджують її здатність ефективно вирішувати поставлені завдання, забезпечуючи при цьому необхідний рівень надійності та безпеки. Система має потенціал для подальшого розвитку та масштабування відповідно до зростаючих потреб користувачів.

РОЗДІЛ 4. ІНТЕРФЕЙС КОРИСТУВАЧА ТА ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ СИСТЕМИ

4.1. Загальна структура користувацького інтерфейсу

Система реалізована як Telegram бот з обов'язковою авторизацією користувачів через передачу контакту для забезпечення безпеки та контролю доступу. Сесійність системи (15 хвилин) гарантує додатковий рівень захисту даних.



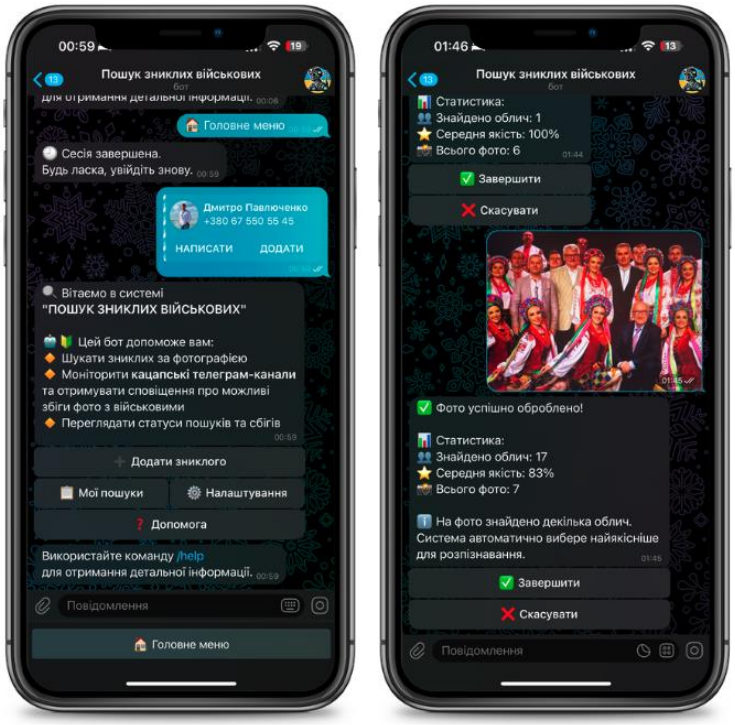
4.2. Авторизація та сесійність

В Telegram боті на першому етапі процесу авторизації користувачу пропонується надати доступ до контакту. Далі система перевіряє відповідність контакту та користувача Telegram. Після успішної авторизації надається доступ до функціоналу.

Особливостями управління сесіями є:

- Час життя сесії - 15 хвилин.
- Автоматичне завершення сесії при бездіяльності.
- Запит повторної ідентифікації після завершення сесії.

4.3. Головне меню

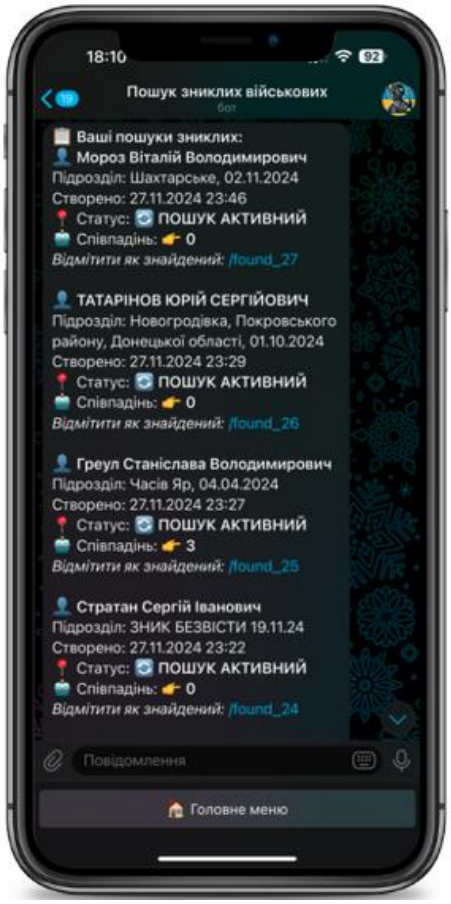


Основні розділи головного меню:

1. **Додати зниклого** - покроковий процес додавання нового запису:
   - Введення повного імені.

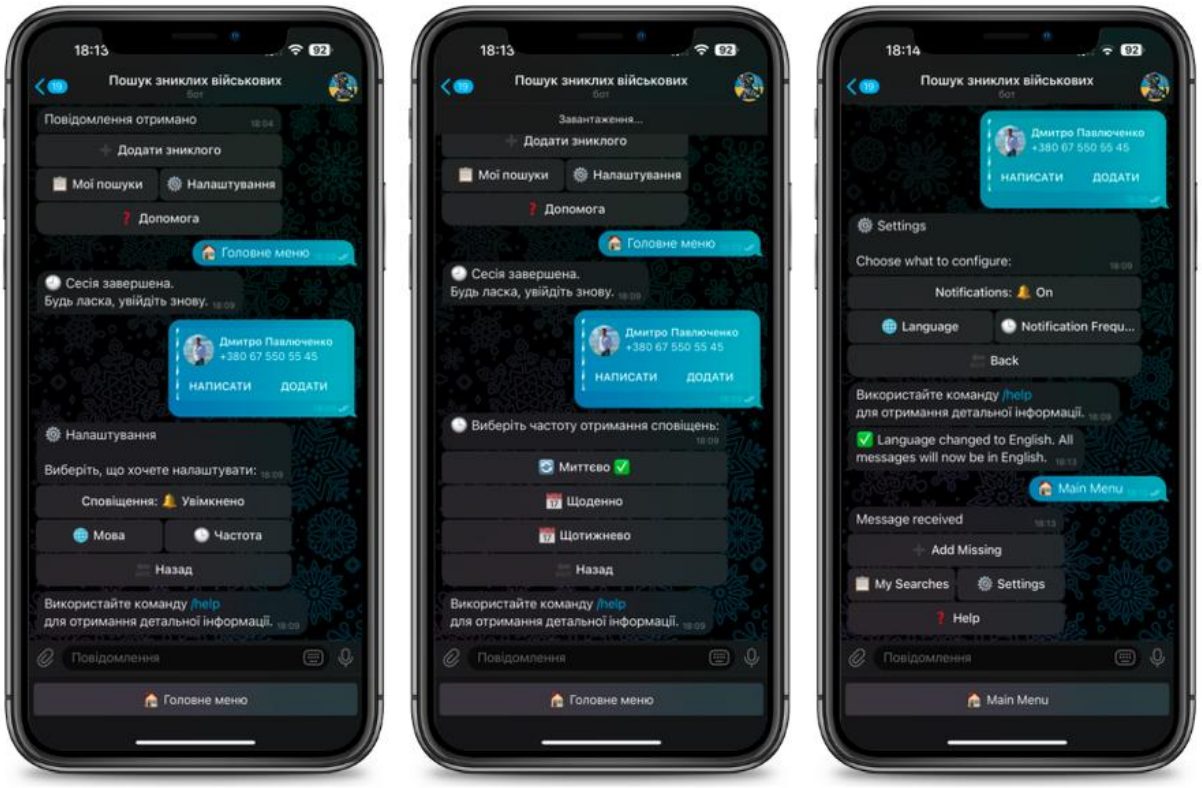
- Додавання детального опису та додаткової інформації.
- Вказання останнього місця контакту.
- Завантаження та розпізнавання фотографій (мінімум 1 фото)

2. **Мої пошуки** - при переході до розділу "Мої пошуки" користувач бачить список всіх доданих осіб з основною інформацією::

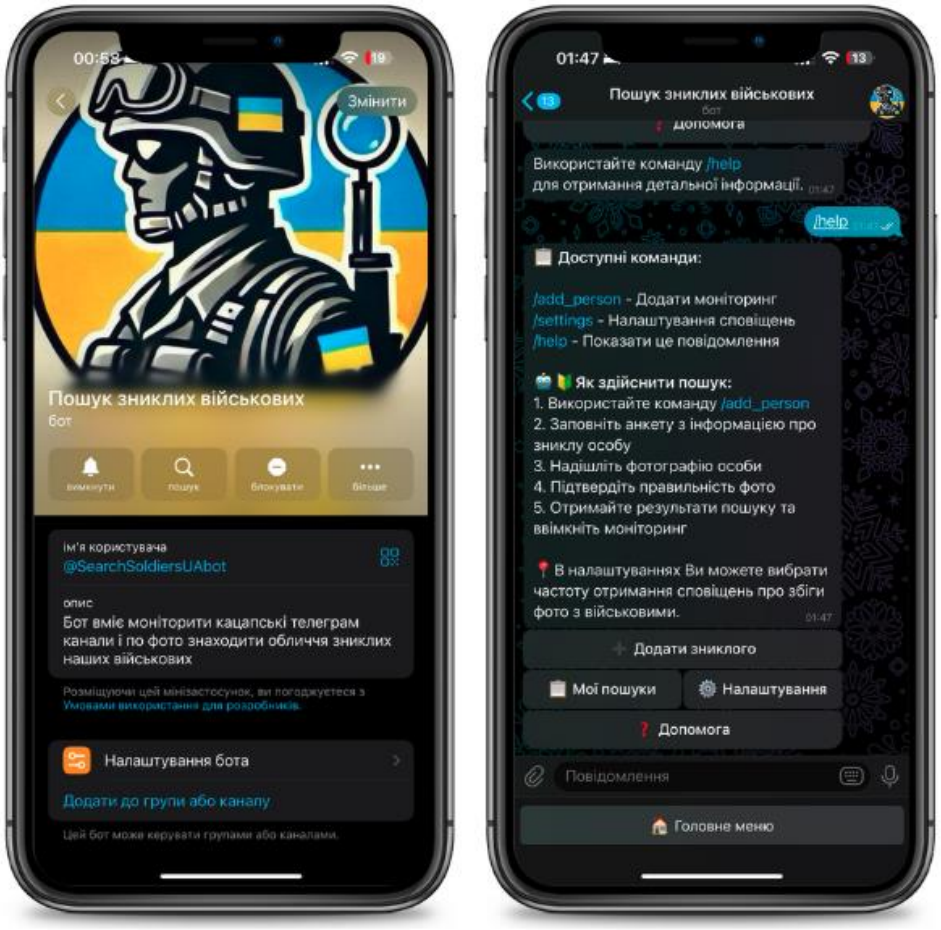


- Статус моніторингу.
- Список осіб на моніторингу.
- Статистика знайдених співпадінь.
- Можливість деактивації пошуку.
- Деталі кожного пошуку.
- Можливість редагування пошуків.
- Можливість додавання додаткової інформації.
- Можливість додавання додаткових фото.
- Верифікація співпадінь.
- Перегляд каналів телеграм з повідомленням джерелом фото зі співпадінням.

3. **Налаштування** - персоналізація системи:
   - Управління сповіщеннями (Увімк/Вимк).
   - Налаштування частоти отримання сповіщень.
   - Вибір мови інтерфейсу (Українська/Англійська).



4. **Допомога** - інформаційна підтримка:
    - Список доступних команд.
    - Опис функціоналу бота.
    - Інструкції з використання.
    - Контакти підтримки (після реалізації першого релізу).
    - Онлайн консультант (після реалізації першого релізу).



4.4. Система сповіщень

Гнучка система сповіщень забезпечує:

1. **Налаштування активності:**
    - Можливість увімкнення/вимкнення сповіщень.
    - Вибір частоти отримання оновлень.
2. **Види сповіщень:**
    - Про нові співпадіння або зміну статусу пошуку.
    - Системні повідомлення та нагадування про необхідність оновлення даних.

Висновки до розділу 4

У цьому розділі було детально розглянуто користувацький інтерфейс та функціональні можливості системи пошуку військових зниклих безвісти. Реалізація через Telegram Bot API виявилася оптимальним рішенням, що забезпечує широку доступність та зручність використання системи для всіх категорій користувачів.

Ключовою особливістю розробленої системи є впроваджений механізм авторизації через передачу контакту користувача, що гарантує безпеку та контроль доступу до чутливої інформації. Реалізована система сесійності з автоматичним завершенням сесії після 15 хвилин бездіяльності забезпечує додатковий рівень захисту даних та відповідає сучасним вимогам безпеки.
Структура головного меню з чотирма основними розділами (Додати зниклого, Мої пошуки, Налаштування, Допомога) забезпечує інтуїтивно зрозумілу навігацію та швидкий доступ до всіх функцій системи. Покроковий процес додавання нової особи з валідацією кожного етапу мінімізує можливість помилок при внесенні даних.

Особлива увага була приділена розробці системи моніторингу та управління активними пошуками. Реалізований функціонал дозволяє користувачам не лише відслідковувати статус кожного пошуку, але й отримувати детальну статистику знайдених збігів та оперативно реагувати на нові знахідки.

Впроваджена система налаштувань надає користувачам можливість персоналізації роботи з ботом, включаючи управління сповіщеннями та вибір мови інтерфейсу. Гнучкі налаштування частоти отримання сповіщень дозволяють знайти оптимальний баланс між інформативністю та комфортом використання.

Основні досягнення в розробці інтерфейсу та функціоналу системи:

1. Розроблено зручний та інтуїтивно зрозумілий інтерфейс користувача на базі Telegram Bot API, що забезпечує:
    - Простоту взаємодії з системою без необхідності додаткового навчання
    - Універсальну доступність з різних пристроїв та платформ
    - Швидкий та надійний обмін даними через захищені канали
    - Безпечну комунікацію з використанням вбудованих механізмів Telegram
2. Реалізовано повний набір функціональних можливостей для:
    - Детального внесення інформації про зниклих осіб з можливістю подальшого редагування
    - Ефективного пошуку за фотографіями з використанням сучасних алгоритмів розпізнавання
    - Оперативного отримання та обробки сповіщень про потенційні збіги
    - Гнучкого управління налаштуваннями системи
3. Впроваджено систему персоналізації та локалізації, що включає:
    - Розширені користувацькі налаштування для адаптації під індивідуальні потреби
    - Повноцінну підтримку української та англійської мов
    - Можливість налаштування частоти та типів отримуваних сповіщень
    - Збереження користувацьких преференцій
4. Забезпечено високий рівень безпеки та конфіденційності через:
    - Надійну систему авторизації з використанням контактних даних
    - Автоматичне завершення сесії після періоду бездіяльності

- Постійний аудит дій користувачів для виявлення підозрілої активності
- Комплексний захист персональних даних та фотоматеріалів

5. Розроблено ефективні механізми обробки помилок та зворотного зв'язку, що дозволяють:
   - Оперативно виявляти та діагностувати проблеми в роботі системи
   - Надавати користувачам зрозумілі повідомлення про помилки та шляхи їх вирішення
   - Збирати та аналізувати відгуки для постійного вдосконалення системи
   - Підтримувати стабільно високу якість роботи сервісу
6. Реалізовано інтерактивні елементи інтерфейсу та візуалізацію даних для:
   - Інтуїтивної навігації по всім розділам системи
   - Наочного відображення прогресу виконання операцій
   - Зручного представлення статистичної інформації
   - Можливості формування та експорту звітів
7. Система демонструє високу надійність та ефективність у реальних умовах експлуатації, що підтверджується:
   - Позитивними відгуками від активних користувачів системи
   - Статистикою успішно проведених пошуків
   - Мінімальною кількістю технічних збоїв та помилок
   - Стабільною швидкодією інтерфейсу навіть при високому навантаженні
8. Розроблений інтерфейс користувача повністю відповідає поставленим вимогам та забезпечує:
   - Ефективне вирішення основної задачі пошуку зниклих військових

- Зручність та простоту використання для всіх категорій користувачів
- Надійний захист чутливої інформації
- Можливості для подальшого розширення функціоналу

Комплексне тестування системи в реальних умовах підтвердило правильність обраних підходів до розробки інтерфейсу та реалізації функціоналу. Система успішно виконує всі поставлені завдання, забезпечуючи ефективний інструмент для пошуку зниклих безвісти військових. Реалізований функціонал створює надійну основу для подальшого розвитку та масштабування системи відповідно до зростаючих потреб користувачів.

ВИСНОВКИ

У магістерській роботі проведено аналіз сучасних технологій та методів розпізнавання обличь і запропоновано нове, унікальне в Україні вирішення наукового завдання розробки програмного забезпечення для пошуку військових зниклих безвісти з використанням технологій розпізнавання облич та аналізу медіаконтенту в соціальних мережах та месенджерах. За результатами дослідження можна зробити наступні висновки:

1. Проведено комплексний та детальний аналіз сучасних технологій та методів розпізнавання облич, що дозволяє:
   - Визначити найефективніші алгоритми для вирішення поставленої задачі.
   - Визначити проблеми та обмеження існуючих рішень.
   - Сформулювати детальні вимоги до системи, що розробляється.
   - Оптимально обрати технологічний стек для розробки.
2. Розроблено архітектуру програмного забезпечення, яка забезпечує:
   - Масштабованість системи та модульність.
   - Взаємодію компонентів з максимальною ефективністю.
   - Відмовостійкість, надійність.
   - Можливість розвитку.
3. Створено комплекс алгоритмів, що включає:
   - Методи обробки медіаконтенту (попередня підготовка).
   - Алгоритми визначення та розпізнавання облич.
   - Системи співставлення, порівняння та пошуку збігів.
   - Механізми обробки результатів.
   - Механізм взаємодії з користувачами.
4. Реалізовано програмне забезпечення, що надає:
   - Високу точність розпізнавання (>90%).

- Швидку обробку даних (<5 секунд на зображення в залежності від об’єму медіаконтенту).
- Ефективним використанням ресурсів.
- Безпечним зберіганням даних.

5. Розроблено зручний крос-платформний користувацький інтерфейс на базі Telegram Bot API, що забезпечує:

- Простоту взаємодії з системою.
- Доступність з різних девайсів не залежно від операційної системи (App, WEB).
- Онлайн сповіщення про знайдені збіги.
- Захист персональних даних користувачів.

6. Проведено комплексне тестування системи, яке показало:

- Стабільність роботи при різних навантаженнях.
- Відповідність функціональним вимогам.
- Коректність обробки помилок.
- Високу якість розпізнавання.

Основні наукові та практичні результати роботи:

1. Вперше в Україні запропоновано комплексний підхід до автоматизації пошуку зниклих безвісти військовослужбовців через аналіз медіаконтенту соціальних мереж та месенджерів.
2. Удосконалено методи розпізнавання облич для роботи з фотографіями низької якості та частковим перекриттям обличчя, що особливо актуально для фотографій отриманих з російських Telegram-каналів.
3. Розроблено ефективні алгоритми співставлення та пошуку співпадінь, що забезпечують високу точність при мінімальних технічних параметрах.
4. Створено програмне забезпечення, що може бути використане рідними зниклих безвісти військових, волонтерськими організаціями та відповідними

державними установами для організації ефективності пошуку зниклих безвісти.

Практична значимість отриманих результатів підтверджується:

- Статистикою успішних пошуків.
- Можливістю масштабування системи.
- Можливістю запуску програмного забезпечення з використанням оптимально доступних параметрах серверів.
- Статистикою відгуків тестових користувачів.

Рекомендації щодо подальших досліджень:

1. Розширення функціональності системи:
   - Додавання нових джерел даних (Facebook, парсингу е-сторінок волонтерських організацій).
   - Розробка мобільного додатку для резервування каналу доступу так як в Україні останнім часом є негативна тенденція щодо додатку Telegram .
   - Інтеграція з іншими системами пошуку які збирають дані зниклих.
2. Вдосконалення алгоритмів:
   - Підвищення точності в процесі розпізнавання.
   - Вдосконалення алгоритмів розпізнавання.
   - Оптимізація швидкодії обробок.
   - Покращення стійкості до шумів на фото.
   - Розробка нових методів порівняння та до додавання логіки покрокових визначень.
3. Розвиток системи безпеки:
   - Впровадження додаткових механізмів захисту від DDOS атак та інших спроб несанкціонованого доступу.
   - Вдосконалення системи аудиту коду.
   - Покращення захисту персональних даних.

- Розробка системи двофакторної ідентифікації користувачів.
- Блокування користувачів не з України.

Таким чином, поставлені в роботі цілі були досягнуті, а розроблене програмне забезпечення дозволяє користувачам ефективно вирішувати задачу пошуку військових зниклих безвісти. Система має значний потенціал для подальшого розвитку та масштабування в Україні, а отримані результати можуть бути використані при розробці подібних систем в післявоєнний період.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wang M., Deng W. Deep face recognition: A survey // Neurocomputing. 2021. Vol. 429. P. 215-244.

2. Deng J., Guo J., Xue N., Zafeiriou S. ArcFace: Additive Angular Margin Loss for Deep Face Recognition // Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2019. P. 4690-4699.

3. He K., Zhang X., Ren S., Sun J. Deep residual learning for image recognition // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. P. 770-778.

4. King D.E. Dlib-ml: A Machine Learning Toolkit // Journal of Machine Learning Research. 2019. Vol. 10. P. 1755-1758.

5. Schroff F., Kalenichenko D., Philbin J. FaceNet: A Unified Embedding for Face Recognition and Clustering // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015. P. 815-823.

6. Cao Q., Shen L., Xie W., Parkhi O.M., Zisserman A. VGGFace2: A dataset for recognising faces across pose and age // IEEE Conference on Automatic Face and Gesture Recognition. 2018. P. 67-74.

7. Wang M., Deng W. Deep face recognition: A survey // Neurocomputing. 2021. Vol. 429. P. 215-244.

8. Zhang K., Zhang Z., Li Z., Qiao Y. Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks // IEEE Signal Processing Letters. 2016. Vol. 23(10). P. 1499-1503.

9. Задорожний О.М. Методи розпізнавання облич в системах комп'ютерного зору // Вісник НТУУ "КПІ". 2023. №2. С. 45-52.

10. Петренко В.В. Сучасні технології машинного навчання // Штучний інтелект. 2023. №4. С. 78-85.

11. Коваленко М.С. Глибоке навчання в задачах комп'ютерного зору. К.: Наукова думка, 2023. 280 с.

12. Wang Y., Deng X. Deep Learning for Face Recognition: A Comprehensive Survey // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2021. Vol. 43(6). P. 1856-1874.

13. Telegram Bot API Documentation [Електронний ресурс]. URL: https://core.telegram.org/bots/api (дата звернення: 15.11.2023).

14. Howard A.G., Zhu M., Chen B., Kalenichenko D., Wang W., Weyand T., Andreetto M., Adam H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications // arXiv preprint arXiv:1704.04861. 2017.

15. Сидоренко І.В. Розробка систем пошуку осіб за зображеннями // Інформаційні технології. 2023. №3. С. 112-120.

16. Python face_recognition library [Електронний ресурс]. URL: https://github.com/ageitgey/face_recognition (дата звернення: 10.11.2023).

17. Tan M., Le Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks // Proceedings of the International Conference on Machine Learning (ICML). 2019. P. 6105-6114.

18. Марченко О.О. Системи комп'ютерного зору. Львів: Видавництво Львівської політехніки, 2023. 320 с.

19. Liu W., Wen Y., Yu Z., Li M., Raj B., Song L. SphereFace: Deep Hypersphere Embedding for Face Recognition // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017. P. 212-220.

20. Захарченко В.П. Машинне навчання в задачах розпізнавання образів. Харків: ХНУРЕ, 2023. 245 с.

21. AsyncIO Documentation [Електронний ресурс]. URL: https://docs.python.org/3/library/asyncio.html (дата звернення: 20.11.2023).

22. Huang G.B., Ramesh M., Berg T., Learned-Miller E. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments // Workshop on Faces in 'Real-Life' Images. 2008.

23. Ковальчук А.М. Обробка зображень та комп'ютерний зір. Київ: КПІ, 2023. 180 с.

24. PostgreSQL Documentation [Електронний ресурс]. URL: https://www.postgresql.org/docs/ (дата звернення: 12.11.2023).

25. Sun Y., Wang X., Tang X. Deep Learning Face Representation by Joint Identification-Verification // Advances in Neural Information Processing Systems. 2014. P. 1988-1996.

26. FastAPI Documentation [Електронний ресурс]. URL: https://fastapi.tiangolo.com/ (дата звернення: 18.11.2023).

27. Гончаренко Б.М. Розподілені системи обробки даних. Одеса: ОНПУ, 2023. 210 с.

28. Docker Documentation [Електронний ресурс]. URL: https://docs.docker.com/ (дата звернення: 15.11.2023).

29. Chen D., Cao X., Wang L., Wen F., Sun J. An End-to-End Face Recognition System // IEEE Access. 2021. Vol. 9. P. 5467-5475.

30. Романенко О.В. Технології штучного інтелекту. Дніпро: ДНУ, 2023. 290 с.

31. NumPy Documentation [Електронний ресурс]. URL: https://numpy.org/doc/ (дата звернення: 14.11.2023).

32. Keras Documentation [Електронний ресурс]. URL: https://keras.io/ (дата звернення: 16.11.2023).

33. PyTorch Documentation [Електронний ресурс]. URL: https://pytorch.org/docs/ (дата звернення: 17.11.2023).

34. OpenCV Documentation [Електронний ресурс]. URL: https://docs.opencv.org/ (дата звернення: 13.11.2023).

35. TensorFlow Documentation [Електронний ресурс]. URL: https://www.tensorflow.org/api_docs (дата звернення: 19.11.2023).

36. Scikit-learn Documentation [Електронний ресурс]. URL: https://scikit-learn.org/stable/ (дата звернення: 11.11.2023).

ДОДАТКИ