

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**До захисту допущено:
В.о. завідувача кафедри
Артем ВОЛОКИТА**

_____ (підпис)

“__” _____ 2026 р.

Дипломний проєкт

на здобуття ступеня бакалавра

**за освітньо-професійною програмою “Інженерія програмного
забезпечення комп’ютерних систем”
спеціальності 121 “Інженерія програмного забезпечення”**

на тему: Інтелектуальна система біометричної автентифікації на основі
антропометрики обличчя

Виконав : студент 4 курсу, групи ІМ-21
(шифр групи)

Крицька Софія Дмитрівна

(прізвище, ім’я, по батькові)

_____ (підпис)

Керівник доц., к.т.н., доц. Селіванов В. Л.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Консультант (нормоконтроль) ст. викл., Коренко Д. В.

(назва розділу)

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

_____ (підпис)

Рецензент _____

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

_____ (підпис)

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

(підпис)

Київ – 2026 р.

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Інженерія програмного забезпечення комп’ютерних систем”

спеціальності 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

Артем ВОЛОКИТА

(підпис)

“ __ ” _____ 2026 р.

ЗАВДАННЯ

на бакалаврський дипломний проєкт студента

Крицької Софії Дмитрівни

1. Тема проєкту Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя

керівник проєкту доц., к.т.н., доц. Селіванов В. Л.

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від __ травня 2026 року № _____

2. Термін здачі студентом закінченого проєкту 04 червня 2026 р.

3. Вихідні дані до проєкту технічна документація, теоретичні дані.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Огляд методів та засобів біометричної автентифікації.

Розділ 2. Обґрунтування вибору технологій та математичний апарат.

Розділ 3. Проектування та архітектура системи.

Розділ 4. Дослідження та аналіз розробленої системи.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) структурна схема системи, функціональна схема (діаграма класів), алгоритм дій програмного забезпечення.

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Коренко Д. В.		

7. Дата видачі завдання «30» серпня 2025 р.

Календарний план

№ п/п	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	<i>Затвердження теми проєкту</i>	<i>10.12.2026-15.12.2026</i>	
2.	<i>Вивчення та аналіз завдання</i>	<i>15.12.2026-15.03.2026</i>	
3.	<i>Розробка архітектури та загальної структури системи</i>	<i>15.03.2026-25.03.2026¹</i>	
4.	<i>Розробка структур окремих підсистем</i>	<i>25.03.2026-5.04.2026</i>	
5.	<i>Програмна реалізація системи</i>	<i>5.04.2026-15.04.2026</i>	
6.	<i>Оформлення пояснювальної записки</i>	<i>15.04.2026-20.05.2026</i>	
7.	<i>Захист програмного продукту</i>	<i>25.04.2026</i>	
8.	<i>Передзахист</i>	<i>23.05.2026</i>	
9.	<i>Захист</i>	<i>18.06.2026</i>	

Студент-дипломник _____ Софія КРИЦЬКА
(підпис)

Керівник проєкту _____ Віктор СЕЛІВАНОВ
(підпис)

АНОТАЦІЯ

У даній бакалаврській роботі було детально розглянуто сучасні методи біометричної автентифікації особи та розроблено локальну інтелектуальну систему верифікації на основі антропометрики обличчя, що функціонує в режимі реального часу. Проаналізовано існуючі нейромережеві архітектури комп'ютерного зору та обґрунтовано вибір конвеєра MediaPipe Face Mesh для високоточної екстракції просторової сітки з 468 ключових точок.

На основі проведеного аналізу було розроблено обчислювальний модуль для векторизованого розрахунку геометричних пропорцій та порівняння ембедінгів за метриками евклідової відстані та косинусної подібності. Створено захищене автономне сховище біометричних шаблонів на базі вбудованої СУБД SQLite із застосуванням бінарної серіалізації масивів та гешування SHA-256 з додаванням криптографічної солі.

Проведено експериментальне дослідження ефективності застосунку із визначенням коефіцієнтів FAR та FRR залежно від зовнішніх умов. Програмний продукт реалізовано мовою Python із застосуванням багатопотокової архітектури на базі фреймворку PyQt.

Ключові слова: біометрична автентифікація, антропометрика обличчя, комп'ютерний зір, MediaPipe Face Mesh, Python.

ANNOTATION

In this project for a Bachelor's Degree, modern methods of biometric user authentication were analyzed in detail, and a local intelligent real-time verification system based on facial anthropometry was developed. Existing computer vision neural network architectures were analyzed, and the selection of the MediaPipe Face Mesh pipeline for the high-precision extraction of a spatial mesh with 468 key points was justified.

Based on the conducted analysis, a computational module was developed for the vectorized calculation of geometric proportions and the comparison of embeddings using Euclidean distance and cosine similarity metrics. A secure, autonomous biometric template repository was created using an embedded SQLite database, employing binary array serialization and SHA-256 hashing with a cryptographic salt.

An experimental study of the application's performance was conducted, determining the FAR and FRR metrics depending on external conditions. The software product was implemented in Python, utilizing a multithreaded architecture based on the PyQt framework.

Keywords: biometric authentication, facial anthropometry, computer vision, MediaPipe Face Mesh, Python.

справки	Формат	Значення			Найменування	Кіл. листів	№ екземп-ляра	Додаток
					Документація загальна			
					Знову розроблена			
	A4	ІАЛЦ.467200.002 ТЗ			Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя	4		
					Технічне завдання			
	A4	ІАЛЦ.467200.003 ПЗ			Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя	113		
					Пояснювальна записка			
	A4	ІАЛЦ.467200.004 Д1			Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя	1		
					Схема структури системи			
	A4	ІАЛЦ.467200.005 Д2			Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя	1		
					Функціональна схема (діаграма класів)			
	A4	ІАЛЦ.467200.006 Д3			Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя	1		
					Послідовність дій програмного забезпечення			
	A4	ІАЛЦ.467200.007 Д4			Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя	1		
					Текст програмного коду			
					ІАЛЦ.467200.001 ОА			
Зм	Лист	№ докум.	Підп	Дата				
Розроб		Крицька С.Д.			Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя Опис альбому	Літ.	Аркуш	Аркушів
Перев		Селіванов В.Л.					1	1
						КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-21		

ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «Інтелектуальна система біометричної автентифікації на основі
антропометрики обличчя»

Київ – 2026

ЗМІСТ

НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ	2
ПІДСТАВИ ДЛЯ РОЗРОБКИ	2
МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ.....	2
ДЖЕРЕЛА РОЗРОБКИ.....	3
ТЕХНІЧНІ ВИМОГИ.....	3
Вимоги до розробленого продукту	3
Вимоги до програмного забезпечення	4
Вимоги до апаратної частини	4
ЕТАПИ РОЗРОБКИ	4

					ІАЛЦ.467200.002 ТЗ			
		№ докум.	Підпис	Дата				
Розробив	Крицька С.Д.				Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя Технічне завдання	Літ.	Аркуш	Аркушів
Перевірив	Селіванов В.Л.						1	4
Реценз.						КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-21		
Н. Контр.	Коренко Д. В.							
Затвердив	Волокита А. М.							

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Повна назва об'єкта розробки — інтелектуальна система біометричної автентифікації на основі антропометрики обличчя. Цей документ встановлює основні вимоги до проєктування, розробки та подальшої експлуатації зазначеного програмного забезпечення.

Цільовою сферою застосування продукту є системи управління доступом та інформаційна безпека. Програма призначена для швидкої та безконтактної верифікації користувачів, що дозволяє надійно захистити цифрові або фізичні ресурси від втручання. Розроблене рішення може використовуватись для безпечного авторизованого входу до локальних робочих станцій, захисту корпоративних даних, а також як інтегрований модуль у комплексах пропуску до закритих приміщень чи для автоматизованого обліку робочого часу.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Офіційною підставою для створення цього програмного продукту є затверджене індивідуальне завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр інженерії програмного забезпечення», який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Головною метою цього проєкту є створення надійної програмної системи, здатної ідентифікувати та верифікувати особу на основі унікальних антропометричних характеристик обличчя. Призначення розробки полягає в

					ІАЛЦ.467200.002 ТЗ	Арк.
						2
Зм.	Арк.	№ докум.	Підпис	Дата		

автоматизації процесу автентифікації користувачів, що дозволить ефективно замінити або доповнити традиційні парольні механізми, підвищивши тим самим загальний рівень захисту інформаційних та фізичних ресурсів від несанкціонованого доступу.

4 ДЖЕРЕЛА РОЗРОБКИ

Інформаційною базою для виконання проекту є сучасна науково-технічна література з комп'ютерного зору та біометрії, актуальні академічні публікації щодо алгоритмів розпізнавання образів, а також офіційна технічна документація до обраного стека технологій та спеціалізованих програмних бібліотек.

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги:

- Простий та інтуїтивно зрозумілий графічний інтерфейс користувача.
- Надати можливість реєстрації нових біометричних профілів шляхом зчитування відеопотоку та збереження еталонних антропометричних даних до локальної бази.
- Забезпечити режим реального часу для розпізнавання та автентифікації осіб перед камерою зі швидким виведенням результату.
- Надати можливість користувачам власноруч налаштовувати базові параметри камери та пороги чутливості алгоритмів верифікації.
- Надати вичерпну та зрозумілу документацію для розробленого продукту.

					ІАЛЦ.467200.002 ТЗ	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

5.2. Вимоги до програмного забезпечення

- ОС Windows, Mac чи Linux.
- Встановлене середовище виконання Python версії 3.8 або вищої.
- Середовище розробки PyCharm, Visual Studio Code або аналогічні.

5.3. Вимоги до апаратної частини

- ЦП не менше ніж Intel Core i3 або відповідний аналог.
- ROM не менше ніж 1 ГБ (для розміщення середовища, бібліотек та файлу бази даних).
- RAM не менше ніж 4 ГБ.
- Вбудована або підключена веб-камера з роздільною здатністю не менше 720p.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми проєкту	10.12.2026-15.12.2026
Вивчення та аналіз завдання	15.12.2026-15.03.2026
Розробка архітектури та загальної структури системи	15.03.2026-25.03.2026 ¹
Розробка структур окремих підсистем	25.03.2026-5.04.2026
Програмна реалізація системи	5.04.2026-15.04.2026
Оформлення пояснювальної записки	15.04.2026-20.05.2026
Захист програмного продукту	25.04.2026

**ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ**

на тему: «Інтелектуальна система біометричної автентифікації на основі
антропометрики обличчя»

Київ – 2026

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ.....	4
ВСТУП	5
РОЗДІЛ 1. ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ БІОМЕТРИЧНОЇ АВТЕНТИФІКАЦІЇ.....	7
1.1 Глобальні поняття та задачі біометрії.....	7
1.1.1 Основні поняття	7
1.1.2 Задачі ідентифікації та верифікації.....	8
1.2 Огляд методів розпізнавання обличчя.....	10
1.2.1 Методи на основі геометричних характеристик та антропометрики обличчя	11
1.2.2 Методи на основі каскадів (HAAR) та нейромереж (CNN)	12
1.3 Аналіз існуючих систем-аналогів	14
1.3.1 Система FaceID	14
1.3.2 Система Luxand.....	17
1.3.3 Бібліотека OpenFace та DeepFace.....	19
1.4 Порівняльний аналіз програмних аналогів та постановка задач	23
ВИСНОВОК ДО РОЗДІЛУ 1	29
РОЗДІЛ 2. ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ТА МАТЕМАТИЧНИЙ АПАРАТ	31
2.1 Детальний огляд алгоритмів, які лягли в основу об'єкта розробки... 31	
2.1.1 Огляд алгоритмів Landmark Detection для виділення 468 ключових точок	31
2.1.2 Математична модель розрахунку векторів ознак (евклідова відстань та косинусна подібність).....	33
2.1.3 Методи попередньої обробки, нормалізації освітлення та вирівнювання обличчя.....	37

					ІАЛЦ.467200.003 ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата				
Розробив	Крицька С.Д.				Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя Пояснювальна записка	Літ.	Аркуш	Аркушів
Перевірив	Селіванов В.Л.						1	113
Реценз.						КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-21		
Н. Контр.	Коренко Д. В.							
Затвердив	Волокита А. М.							

2.2 Мова програмування.....	39
2.2.1 Мова Python	39
2.2.2 Переваги мови Python для AI-задач та обробки даних	40
2.2.3 Аналіз альтернатив (Java/C++) та їх недоліки для поточної задачі	41
2.2.4 Підсумок щодо вибору мови локального середовища.....	42
2.3 Технології та бібліотеки	43
2.3.1 OpenCV (обробка відеопотоку)	43
2.3.2 MediaPipe (модель Face Mesh)	45
2.3.3 NumPy (матричні обчислення)	48
2.3.4 PyQt (для графічного інтерфейсу).....	50
2.3.5 SQLite (для локальної бази даних еталонів)	52
2.3.6 Pickle та Hashlib (засоби серіалізації та безпеки даних)	52
ВИСНОВОК ДО РОЗДІЛУ 2	55
РОЗДІЛ 3. ПРОЄКТУВАННЯ ТА ДЕТАЛІ РОЗРОБКИ ПРОГРАМНОГО КОМПЛЕКСУ	58
3.1 Розробка програмних компонентів обчислювального ядра	58
3.1.1 Клас FaceMeshExtractor	58
3.1.2. Клас ImagePreprocessor.....	61
3.1.3. Клас AnthropometryAnalyzer	64
3.1.4. Клас BiometricMatcher	68
3.1.5. Клас SecurityManager.....	71
3.1.6. Клас TextureLivenessDetector	73
3.1.7. Клас LivenessDetector	75
3.2. Проєктні рішення підсистеми збереження даних.....	77
3.2.1. Клас DatabaseManager	77
3.2.2. Фізичне проєктування структури таблиць Users та AuthLog	80
3.3. Реалізація компонентів користувацького інтерфейсу	83
3.3.1. Патерн асинхронного виконання на базі класу VideoThread	84

3.3.2. Клас MainWindow	87
ВИСНОВОК ДО РОЗДІЛУ 3	91
РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ	93
4.1 Огляд графічного інтерфейсу користувача	93
4.2 Функціонування локальної бази даних та захист інформації	96
4.3 Експериментальне дослідження точності біометричної автентифікації.....	98
4.4 Аналіз продуктивності та швидкодії обчислювального ядра	100
4.5 Комплексне тестування програмного забезпечення	103
4.6 Рекомендації щодо подальшого розвитку та вдосконалення системи.....	106
ВИСНОВОК ДО РОЗДІЛУ 4	108
ВИСНОВКИ.....	109
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	111
ДОДАТОК 1	3
ДОДАТОК 2.....	5
ДОДАТОК 3.....	7
ДОДАТОК 4.....	10

ПЕРЕЛІК СКОРОЧЕНЬ

AAM	(Active Appearance Models) Активні моделі зовнішнього вигляду
API	(Application Programming Interface) Інтерфейс програмування застосунків
ASM	(Active Shape Models) Активні моделі форми
CNN	(Convolutional Neural Network) Згортоква нейронна мережа
CPU	(Central Processing Unit) Центральний процесор
FAR	(False Acceptance Rate) Коефіцієнт помилкового допуску
FPS	(Frames Per Second) Кількість кадрів на секунду
FRR	(False Rejection Rate) Коефіцієнт помилкової відмови
GPU	(Graphics Processing Unit) Графічний процесор
HAAR	(Haar-like features) Ознаки, подібні до вейвлетів Хаара
IR	(Infrared) Інфрачервоне випромінювання (інфрачервоний сенсор)
LBP	(Local Binary Patterns) Локальні бінарні шаблони
LFW	(Labeled Faces in the Wild) База даних еталонних зображень облич
PCA	(Principal Component Analysis) Метод головних компонентів
PIN	(Personal Identification Number) Персональний ідентифікаційний номер
RFID	(Radio Frequency Identification) Радіочастотна ідентифікація
RGB	(Red, Green, Blue) Стандартна кольорова модель матриці камери
ROI	(Region of Interest) Регіон інтересу на зображенні
SDK	(Software Development Kit) Набір засобів розробки програмного забезпечення
СКУД	Система контролю та управління доступом
СУБД	Система управління базами даних

ВСТУП

Значна частина науково-технічних задач, що виникають у сучасній прикладній математиці, інформаційній безпеці та системах штучного інтелекту, безпосередньо пов'язана з проблемою побудови стійких і надійних механізмів контролю доступу. Процес проектування захищених програмних комплексів ускладнюється специфічними властивостями:

- суттєва нелінійність та стохастичність вхідних інформаційних потоків;
- висока розмірність простору ознак, що описують об'єкт розпізнавання;
- чутливість математичних моделей до деструктивного впливу зовнішнього середовища (інтенсивність світла, зміна просторових ракурсів);
- вразливість обчислювальних алгоритмів до випадкових шумів, дисторсії та артефактів цифровізації;
- неможливість опису складних біометричних паттернів за допомогою простих лінійних аналітичних функцій;
- значна обчислювальна складність процесів сегментації, локалізації та трекінгу структурних елементів об'єкта;
- жорсткі обмеження щодо часових затримок (числа кадрів на секунду) при локальній обробці даних;
- багатовимірна топологія розподілу цифрових шаблонів та векторних вбудовувань у координатному просторі.

Протягом останніх років у галузі комп'ютерного зору сформувався окремий напрям інтелектуального аналізу даних. У межах даної роботи бакалавра досліджуються та інтегруються математичні підходи й програмні архітектури, які в науковій періодиці класифікують як антропометричні сітки, геометричні матриці ознак, згорткові нейромережеві контури, регресійні детектори та дескрипторні екстрактори.

					ІАЛЦ.467200.003 ПЗ	Арк.
						5
Зм.	Арк.	№ докум.	Підпис	Дата		

Фундаментальна база та сучасний інструментарій для роботи з подібними системами розроблені переважно англомовними дослідниками. Через це у практиці розробки частіше фігурують поняття «facial landmarks» та «embeddings». Для термінологічної уніфікації та дотримання вітчизняних академічних стандартів, у межах цього проєкту дані дефініції інтерпретовано як «антропометричні ключові точки» та «біометричні векторні шаблони».

Біометричні системи ідентифікації за геометрією обличчя можна диференціювати за декількома архітектурними та алгоритмічними ознаками. Доцільно виділити такі базові категорії:

- структурно-геометричні підходи, орієнтовані на обчислення базових просторових інтервалів;
- частотно-текстурні методи та статистичні дискримінанти (зокрема, каскадні класифікатори та локальні бінарні шаблони);
- інтелектуальні технології Landmark Detection на базі щільних деформаційних моделей;
- глибокі згорткові нейромережеві архітектури (CNN) для формування релевантних ознак.

Варто відзначити, що базові концепції детекції об'єктів, як-от каскади Хаара або стандартні згорткові шари, є глибоко вивченими та широко представленими у вітчизняних підручниках і дисертаціях. Натомість, у даному дипломному проєкті фокус зміщено на практичне створення та дослідження автономного обчислювального ядра на основі передових високовізуальних моделей (а саме технології Face Mesh на 468 точок координатної сітки) у поєднанні з реляційними механізмами локального менеджменту шаблонів, що є малодослідженим рішенням у контексті розробки ізольованого десктопного програмного забезпечення.

					ІАЛЦ.467200.003 ПЗ	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ БІОМЕТРИЧНОЇ АВТЕНТИФІКАЦІЇ

1.1 Глобальні поняття та задачі біометрії

Інтеграція засобів штучного інтелекту в системи контролю та управління доступом (СКУД) трансформувала класичні підходи до безпеки. Сучасні програмні комплекси поступово відмовляються від традиційної парольної автентифікації на користь біометричних технологій, які базуються на аналізі унікальних характеристик людини [3, 9].

1.1.1 Основні поняття

У контексті інформаційних технологій та інженерії програмного забезпечення, біометрична система — це спеціалізований апаратно-програмний комплекс, призначений для розпізнавання особи на основі математичного аналізу її фізіологічних або поведінкових рис.

Усі біометричні ознаки, що піддаються комп'ютерній обробці, класифікують на дві великі групи:

1. Статичні (фізіологічні) ознаки — невід'ємні фізичні характеристики людини, які формуються на етапі ембріонального розвитку і залишаються відносно незмінними протягом життя. До них належать: геометрія обличчя, папілярний візерунок пальців, структура райдужної оболонки та сітківки ока, термограма обличчя, геометрія долоні тощо [1, 3].
2. Динамічні (поведінкові) ознаки — характеристики, що формуються на основі підсвідомих рухів або поведінкових патернів під час виконання певних дій. До цієї групи відносять: клавіатурний почерк (динаміку

					ІАЛЦ.467200.003 ПЗ	Арк.
						7
Зм.	Арк.	№ докум.	Підпис	Дата		

натискання клавіш), ходу, динаміку підпису та голосові характеристики [1, 9].

Для того щоб певна біометрична характеристика (зокрема, антропометрика обличчя) могла ефективно використовуватися в інженерних системах захисту, вона має відповідати низці фундаментальних критеріїв:

- Універсальність (Universality) — наявність даної ознаки у переважній більшості потенційних користувачів системи.
- Унікальність (Uniqueness) — високий ступінь індивідуальності ознаки, що дозволяє диференціювати двох будь-яких людей за допомогою математичних алгоритмів.
- Незмінність (Permanence) — стійкість метрик до впливу часу та вікових змін.
- Вимірюваність (Collectability) — можливість кількісного вимірювання ознаки за допомогою доступних сенсорів (наприклад, звичайної веб-камери) та її подальшої цифровізації.

Основним концептом у розробці таких систем є біометричний шаблон (еталон). Це цифрова математична модель (у сучасних системах — масив даних або n -вимірний вектор ознак, так званий ембедінг), яка екстрагується з первинного біометричного зразка під час реєстрації користувача (Enrollment) і зберігається в базі даних для подальшого порівняння. Важливо зазначити, що з міркувань кібербезпеки зберігається не саме фотографічне зображення, а лише згенерований хеш або векторний еквівалент, який неможливо реверсивно перетворити у вихідне зображення [2, 3].

1.1.2 Задачі ідентифікації та верифікації

Функціонування будь-якої біометричної системи зводиться до вирішення однієї з двох базових алгоритмічних задач: ідентифікації або верифікації [9].

					ІАЛЦ.467200.003 ПЗ	Арк.
						8
Зм.	Арк.	№ докум.	Підпис	Дата		

Попри те, що в повсякденній мові ці терміни часто ототожнюють, з точки зору обчислювальної складності та архітектури програмного забезпечення це принципово різні процеси.

Верифікація (Verification) — це процес підтвердження заявленої особистості. У цьому сценарії користувач спочатку заявляє про себе системі (наприклад, вводить логін, прикладає смарт-картку або обирає свій профіль), після чого система порівнює його поточний біометричний зразок виключно з одним еталоном, що відповідає цьому ідентифікатору.

Математична суть верифікації — задача порівняння типу «один-до-одного» (1:1). Обчислювальна складність $O(1)$ відносно розміру бази даних. Система розраховує міру подібності (наприклад, евклідову відстань) між двома векторами і приймає бінарне рішення (True/False) на основі наперед заданого порогу чутливості (Threshold). Сферою застосування є локальні системи контролю доступу до комп'ютерів, розблокування смартфонів. Саме цей підхід є базовим для розроблюваної в даній дипломній роботі локальної системи.

Ідентифікація (Identification) — це процес розпізнавання невідомої особи серед множини всіх зареєстрованих користувачів. Користувач не надає жодної попередньої інформації про себе. Система повинна просканувати всю базу біометричних шаблонів, щоб знайти збіг або констатувати його відсутність.

Математична суть ідентифікації — це задача пошуку типу «один-до-багатьох» (1:N), де N — кількість записів у базі даних. Обчислювальна складність $O(N)$ при лінійному пошуку. У великих базах даних це вимагає застосування складних структур даних (наприклад, KD-дерев) та методів кластеризації для оптимізації швидкодії. Сферою застосування є системи відеоспостереження у місцях масового скупчення людей, криміналістика, глобальні державні реєстри.

Надійність роботи алгоритмів як верифікації, так і ідентифікації не є абсолютною через стохастичну природу вхідних даних (різне освітлення, шум матриці камери). Тому якість розроблених систем в інженерії оцінюється

					ІАЛЦ.467200.003 ПЗ	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

статистичними метриками: коефіцієнтом помилкового допуску (FAR — False Acceptance Rate) та коефіцієнтом помилкової відмови (FRR — False Rejection Rate), мінімізація яких є головним завданням при проектуванні обчислювального ядра [3, 9].

1.2 Огляд методів розпізнавання обличчя

Задачі комп'ютерного зору, пов'язані з автоматичним розпізнаванням обличчя, пройшли тривалий шлях еволюції від простих евристичних підходів до складних математичних моделей на основі глибокого машинного навчання [4, 16]. З точки зору інженерії програмного забезпечення, будь-який алгоритм розпізнавання обличчя концептуально складається з двох послідовних етапів: детекції (знаходження обличчя на зображенні та виділення регіону інтересу — ROI) та екстракції ознак (перетворення піксельної інформації у компактний математичний вектор для подальшого порівняння).

Враховуючи високу обчислювальну складність обробки растрових зображень, дослідниками було запропоновано десятки різноманітних алгоритмів, які можна класифікувати за способом формування простору ознак. До основних класів належать глобальні методи (аналіз зображення в цілому, наприклад, метод головних компонентів — PCA), структурні або локальні методи (аналіз окремих рис та їх взаємного розташування), а також гібридні підходи. Для обґрунтування архітектури нашої системи детально розглянемо методи, що базуються на геометричних параметрах, та сучасні підходи на основі машинного навчання.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		10

1.2.1 Методи на основі геометричних характеристик та антропометрики обличчя

Методи розпізнавання на основі геометричних характеристик історично були одними з перших підходів у біометрії, оскільки вони найбільш наближені до того, як людина підсвідомо ідентифікує інших людей. В основі цих методів лежить математичний апарат антропометрики — науки про вимірювання розмірів та пропорцій людського тіла (в даному випадку — обличчя) [1, 5].

Процес екстракції ознак у геометричних методах зводиться до локалізації на зображенні набору так званих фідуціальних (антропометричних) точок. До ключових точок традиційно відносять куточки очей, центри зіниць, кінчик та основу носа, куточки рота, лінію підборіддя та брови. Після локалізації цих точок алгоритм розраховує структурний вектор ознак, який формується з таких геометричних примітивів:

- Евклідові відстані між заданими парами точок (наприклад, міжзінична відстань, відстань від носа до губ);
- Кутові міри (кути, утворені відрізками, що з'єднують певні точки);
- Відносні пропорції (відношення однієї відстані до іншої), що робить систему частково інваріантною до масштабування зображення (віддалення чи наближення користувача до камери).

Класичними алгоритмами, що реалізують цей підхід, є активні моделі форми (Active Shape Models, ASM) та активні моделі зовнішнього вигляду (Active Appearance Models, AAM). Вони використовують статистичні моделі деформації точкових сіток, які ітеративно "натягуються" на обличчя у кадрі, намагаючись мінімізувати функцію енергії між моделлю та контурами на зображенні [4].

Головною перевагою суто геометричних методів є висока швидкість порівняння: зіставлення двох математичних векторів, що містять лише кілька

десятьків числових значень, виконується за долі мілісекунди, що ідеально підходить для локальних пристроїв з обмеженими обчислювальними ресурсами. Крім того, такі моделі потребують мінімального обсягу пам'яті для зберігання бази еталонів. Однак класичні геометричні методи виявилися критично вразливими до зміни міміки, ракурсу зйомки (повороту голови) та часткових перекриттів (оклюзій). Саме тому в сучасних інтелектуальних системах чиста геометрія використовується не ізольовано, а як фінальний етап (метричний класифікатор) після того, як ключові точки були високоточно знайдені за допомогою нейронних мереж.

1.2.2 Методи на основі каскадів (HAAR) та нейромереж (CNN)

З розвитком обчислювальних потужностей фокус змістився на алгоритми, здатні самостійно формувати репрезентативні ознаки на основі статистичного аналізу текстури зображення.

1. Метод Віолі-Джонса (каскади Хаара)

Одним із найважливіших проривів у комп'ютерному зорі став алгоритм детекції, запропонований Полом Віолою та Майклом Джонсом у 2001 році. Він базується на використанні ознак, подібних до вейвлетів Хаара (Haar-like features) — простих прямокутних фільтрів, що обчислюють різницю сум пікселів у світлих і темних областях зображення (наприклад, область очей зазвичай темніша за перенісся) [10].

Для забезпечення роботи в реальному часі алгоритм застосовує концепцію «інтегрального зображення», яка зводить обчислення суми пікселів у будь-якому прямокутнику до чотирьох операцій звернення до пам'яті. Далі алгоритм використовує метод машинного навчання AdaBoost для вибору найбільш релевантних ознак з тисяч можливих, формуючи «каскад класифікаторів». Каскад працює за принципом відбраковування: прості класифікатори швидко відхиляють ділянки фону, дозволяючи алгоритму

					ІАЛЦ.467200.003 ПЗ	Арк.
						12
Зм.	Арк.	№ докум.	Підпис	Дата		

фокусувати ресурси лише на регіонах, де ймовірно є обличчя. Незважаючи на безпрецедентну швидкодію, метод Хаара має серйозні недоліки: він чутливий до поворотів голови (працює переважно в анфас) та зміни освітленості, а також непридатний для точної ідентифікації чи екстракції дрібних антропометричних точок [16].

2. Методи на основі згорткових нейронних мереж (CNN)

Сучасним State-of-the-Art (передовим) підходом у біометрії є використання глибокого машинного навчання, зокрема згорткових нейронних мереж (Convolutional Neural Networks, CNN). На відміну від класичних методів, де інженер вручну задає правила пошуку ознак (ручне конструювання ознак), CNN здатні автоматично вивчати ієрархічні представлення даних під час тренування на мільйонах зображень.

Архітектура CNN складається з каскаду згорткових шарів (Convolutional layers), що застосовують матричні фільтри для виділення локальних патернів (граней, текстур, контурів), та шарів субдискретизації (Pooling layers) для зменшення розмірності простору та забезпечення інваріантності до дрібних зсувів. Глибокі мережі (наприклад, архітектури ResNet, MobileNet) перетворюють вхідне зображення обличчя на щільний вектор високої розмірності у неевклідовому просторі — так званий біометричний ембедінг (Embedding) [11, 15].

Головна перевага CNN-моделей полягає в їхній надвисокій точності та стійкості до нелінійних шумів, різного освітлення, наявності окулярів або зміни міміки. Однак, традиційні великі нейромережі є надзвичайно ресурсомісткими та вимагають наявності графічних прискорювачів (GPU), що ускладнює їх інтеграцію в автономні локальні системи. Компромісним рішенням у сучасній інженерії ПЗ є використання полегшених (lightweight) згорткових архітектур як регресійних детекторів ключових точок. Вони застосовують потужність CNN не для генерації абстрактних векторів текстури, а для надточного прогнозування декартових координат сотень антропометричних точок на

					ІАЛЦ.467200.003 ПЗ	Арк.
						13
Зм.	Арк.	№ докум.	Підпис	Дата		

обличчі. Такий гібридний підхід об'єднує швидкість та логіку математичної геометрії зі стійкістю та точністю глибоких нейромереж [12].

1.3 Аналіз існуючих систем-аналогів

Невід'ємним етапом проектування будь-якого програмного забезпечення є етап системного аналізу існуючих на ринку рішень. У сфері біометричної автентифікації за геометрією обличчя представлено широкий спектр продуктів: від закритих апаратно-програмних комплексів корпоративного рівня до відкритих наукових фреймворків (Open Source). Для обґрунтування архітектури та технологічного стека розроблюваної локальної системи було проведено детальне дослідження найбільш релевантних аналогів.

1.3.1 Система FaceID

Face ID — це пропрієтарна апаратно-програмна система біометричної автентифікації, розроблена компанією Apple. На відміну від більшості програмних рішень, що аналізують 2D-зображення звичайної RGB-камери, Face ID спирається на спеціалізований апаратний модуль камери TrueDepth.

З інженерної точки зору, процес автентифікації у Face ID побудований на основі методів 3D-антропометрики та оптичної топографії поверхні обличчя. Система використовує інфрачервоний випромінювач (Flood Illuminator) для підсвічування обличчя незалежно від умов зовнішнього освітлення, після чого точковий проектор (Dot Projector) за допомогою дифракційних оптичних елементів проектує на обличчя матрицю, що складається з понад 30 000 невидимих інфрачервоних точок. Інфрачервона камера зчитує цей точковий патерн, який зазнає просторових деформацій під впливом рельєфу обличчя.

					ІАЛЦ.467200.003 ПЗ	Арк.
						14
Зм.	Арк.	№ докум.	Підпис	Дата		

Загальну схему архітектури апаратних компонентів TrueDepth та принцип взаємодії зі спектральними датчиками наведено нижче на рисунку 1.1.



Рисунок 1.1 — Схема оптичного тракту камери TrueDepth та формування структурованого інфрачервоного патерна в системі FaceID

Аналіз представленого архітектурно-оптичного рішення дозволяє деталізувати фізико-математичні принципи, покладені в основу технології структурованого світла (Structured Light Technology). Використання інфрачервоного спектра з довжиною хвилі (орієнтовно 940 нм), що лежить поза межами сприйняття людського ока, забезпечує високу інваріантність системи до умов зовнішнього природного чи штучного освітлення. Це нівелює проблему затінення або надмірного експонування (засвітів), яка є критичною для класичних RGB-камер.

Точковий проектор створює регулярну квазівипадкову матрицю точок. Потрапляючи на об'ємний об'єкт (обличчя користувача), координати точок зміщуються відносно свого початкового (еталонного) положення на площині.

Інфрачервона камера фіксує відбитий патерн у вигляді цифрового кадру з високою контрастністю. Спеціалізований алгоритм комп'ютерного зору обчислює величину зміщення кожної точки за допомогою методів оптичної триангуляції.

На основі отриманих векторів зміщень нейронний співпроцесор (Neural Engine) у реальному часі реконструює тривимірну хмару точок (Point Cloud),

					ІАЛЦ.467200.003 ПЗ	Арк.
						15
Зм.	Арк.	№ докум.	Підпис	Дата		

яка математично описує унікальні антропометричні пропорції обличчя (відстань між зіницями, глибину посадки очей, форму носа, лінію підборіддя тощо).

Згенерована 3D-модель перетворюється на нормалізований математичний біометричний шаблон (embedding), який за допомогою захищеного протоколу передається до ізольованого криптографічного анклаву процесора (Secure Enclave) для фінального порівняння з еталоном.

Перевагою такого підходу є безпрецедентно високий рівень безпеки та стійкості до фальсифікацій. Заявлений компанією коефіцієнт помилкового доступу (FAR) становить 1 до 1 000 000. Система апаратно захищена від атак підміни (spoofing) за допомогою роздрукованих фотографій чи навіть високоякісних силіконових 3D-масок завдяки вбудованому алгоритму liveness detection (перевірці на життєздатність особи за допомогою аналізу мікрорухів та рельєфу) [3].

Проте, в контексті задачі проектування, дана архітектура має суттєві недоліки. Описана система є абсолютно закритою (black-box), жорстко прив'язаною до пропрієтарної екосистеми iOS та специфічного дороговартісного обладнання (IR-сенсорів, лазерних проекторів). Її архітектурні та схемотехнічні рішення неможливо портувати для захисту звичайних персональних комп'ютерів, ноутбуків або кросплатформних десктопних застосунків, які у більшості випадків оснащені стандартними інтегрованими веб-камерами.

Це обмеження підтверджує наукову та практичну доцільність, а також актуальність розробки інтелектуальних софтверних (програмних) систем біометричної автентифікації. Такі системи повинні базуватися на сучасних архітектурах глибоких нейронних мереж для виділення просторових антропометричних маркерів із класичних 2D-кадрів (RGB-відеопотоку), забезпечуючи високу точність верифікації без залучення спеціалізованих апаратних сенсорів глибини.

					ІАЛЦ.467200.003 ПЗ	Арк.
						16
Зм.	Арк.	№ докум.	Підпис	Дата		

1.3.2 Система Luxand

Luxand FaceSDK — це високопродуктивне комерційне програмне забезпечення (Software Development Kit) корпоративного класу, призначене для інтеграції функцій комп'ютерного зору та розпізнавання обличчя у сторонні прикладні застосунки. Дана бібліотека розроблена з урахуванням кросплатформної сумісності та підтримує велику множину мов програмування високого рівня (C++, C#, Java, Python), а також функціонує в архітектурах поширених операційних систем (Windows, Linux, macOS, iOS, Android).

В основі алгоритмічного комплексу Luxand лежить гібридний підхід, що поєднує класичні статистичні методи аналізу зображень (такі як Active Shape Models або Active Appearance Models) із сучасними глибокими згортковими нейронними мережами (CNN). Цей математичний апарат орієнтований на динамічну екстракцію понад 70 ключових антропометричних точок, які описують геометрію та просторову конфігурацію критичних зон обличчя: контури очей, брів, ліній губ, периметр підборіддя та морфологічні особливості форми носа. Характерний приклад розподілу та фіксації реперних точок алгоритмами FaceSDK у реальному часі наведено нижче на рисунку 1.2.



Рисунок 1.2 — Графічне представлення схеми локалізації та трекінгу антропометричних ключових точок у пропрієтарній системі Luxand FaceSDK

Детальний аналіз наведеної схеми антропометричного дозволяє стверджувати, що алгоритм FaceSDK виконує просторову нормалізацію обличчя перед безпосереднім етапом порівняння. За рахунок фіксації стабільних біометричних констант, система здатна будувати інваріантний математичний шаблон (embedding) навіть у складних умовах експлуатації.

Програмні фільтри здатні апроксимувати положення прихованих точок у випадку перекриття обличчя медичними масками, шарфами або оптичними окулярами. Якщо частина точок (наприклад, зона рота) стає недоступною для детекції, нейромережевий класифікатор перераховує ваги вагових коефіцієнтів, фокусуючись на відкритих зонах (периорбітальна область, лобна кістка).

Система зберігає працездатність при значних кутах повороту та нахилу голови — до 35 градусів у всіх трьох площинах (pitch, yaw, roll). Для цього застосовуються алгоритми афінних перетворень, які віртуально розгортають тривимірну проекцію обличчя у фронтальну площину (методи деформації текстур або компенсації ракурсу).

Суттєвою перевагою Luxand FaceSDK з точки зору системної інженерії є її висока швидкодія та низька латентність обчислювальних процесів. Ядро бібліотеки оптимізовано на низькому рівні під багатопотокові обчислення та архітектурні інструкції сучасних процесорів (наприклад, підтримка векторних розширень SIMD, AVX/AVX2). Наявність детально документованого та стабільного прикладного інтерфейсу програмування (API) дозволяє розробникам оперативно впроваджувати готові модулі автентифікації, які демонструють надійні показники точності навіть при роботі зі звичайними поточковими веб-камерами низької роздільної здатності (наприклад, 0.3 Мп або стандартні 720p сенсори) без необхідності додаткового апаратного підсвічування [5].

Незважаючи на високі експлуатаційні характеристики, в контексті вирішення конкретної інженерної задачі проектування нашої інтелектуальної системи, Luxand FaceSDK має низку критичних недоліків:

					ІАЛЦ.467200.003 ПЗ	Арк.
						18
Зм.	Арк.	№ докум.	Підпис	Дата		

- Закритість архітектурних рішень (Black-box модель).
- Суворе комерційне ліцензування.

Повністю закритий вихідний код та відсутність доступу до внутрішнього математичного апарату нейронної мережі унеможливлюють проведення тонкого налаштування, модифікації шарів мережі або оптимізації функцій втрат (loss functions) під специфічні науково-дослідні чи інженерні завдання нашого проєкту.

Пропрієтарний статус продукту та висока вартість ліцензії для розгортання в промислових масштабах роблять його економічно недоцільним для локальних некомерційних систем, стартапів на ранніх етапах або академічних дослідницьких проєктів

Таким чином, детальний огляд системи Luxand підтверджує доцільність створення альтернативного, гнучкого софтверного рішення. Проєктована система повинна поєднувати високу точність антропометричного аналізу (зокрема, шляхом переходу до більш щільних просторових моделей, таких як MediaPipe Face Mesh, що оперує 468 точками) із повною автономністю, відкритістю архітектури та відсутністю залежності від ліцензійних обмежень сторонніх комерційних постачальників.

1.3.3 Бібліотека OpenFace та DeepFace

У сфері академічних досліджень, прототипування та розробки відкритого програмного забезпечення (Open Source) стандартами де-факто є сучасні розгалужені фреймворки комп'ютерного зору OpenFace та DeepFace.

OpenFace — це спеціалізована бібліотека, орієнтована на детекцію, вирівнювання та розпізнавання облич з використанням глибоких нейронних мереж (DNN). Вона спроектована на базі мобільно-оптимізованої архітектури FaceNet від компанії Google. Математична суть базового алгоритму полягає в

тому, що згорткова нейронна мережа (CNN) виконує нелінійне відображення цифрового кадру у низьковимірний компактний простір ознак, генеруючи 128-вимірний або 512-вимірний вектор (ембедінг). Цей вектор лежить на одиничній гіперсфері, де евклідова відстань (або косинусна подібність) між двома точками прямо пропорційна візуальній схожості вихідних облич.

DeerFace — це високорівневий гібридний фреймворк для мови програмування Python, який абстрагує та об'єднує під єдиним уніфікованим інтерфейсом прикладного програмування (API) найпотужніші State-of-the-Art (SOTA) моделі розпізнавання: VGG-Face, Google FaceNet, Facebook DeepFace, DeepID, ArcFace та Dlib. Головна перевага фреймворку полягає в архітектурній можливості виконання повного, наскрізного обчислювального пайплайну біометричної верифікації. Узагальнену структурно-функціональну схему цього софтверного пайплайну обробки вхідних даних наведено нижче на рисунку 1.3.

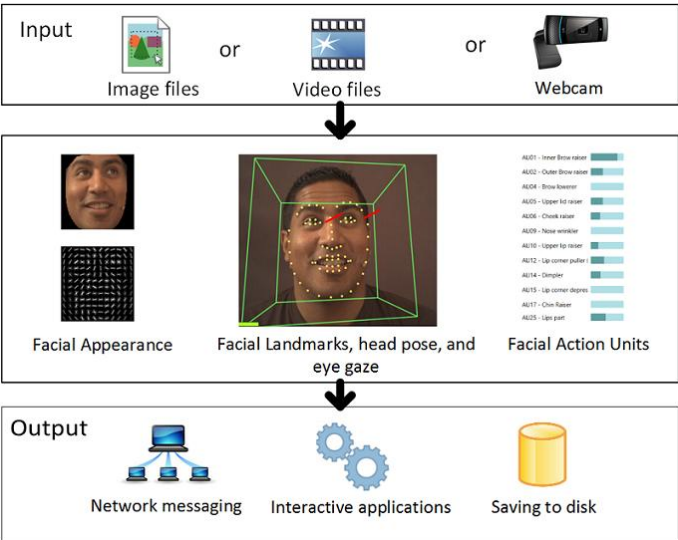


Рисунок 1.3 — Структурно-функціональна схема софтверного пайплайну верифікації та екстракції ознак у фреймворках OpenFace та DeerFace

Детальний аналіз наведеної архітектури софтверного тракту дозволяє формалізувати послідовність етапів трансформації біометричних даних, що протікають у системі «з коробки»:

1. Детекція та локалізація (Face Detection).

Первинний модуль (на базі алгоритмів MTCNN, RetinaFace або Haar Cascades) ізолює обличчя користувача від фонового шуму, визначаючи обмежувальну рамку (bounding box).

2. Просторова нормалізація та афінні перетворення (Face Alignment).

За допомогою реперних точок (наприклад, центрів зіниць та кінчика носа) алгоритм обчислює матрицю двовимірного афінного перетворення. Це дозволяє віртуально повернути, масштабувати та центрувати обличчя користувача, компенсуючи нахили голови за трьома осями (yaw, pitch, roll). Наявність цього етапу є критично важливою, оскільки згорткові шари нейромереж є чутливими до просторових деформацій і розворотів об'єкта в кадрі.

3. Генерація дескрипторів (Feature Extraction).

Нормалізований фрагмент зображення подається на вхід глибокої мережі, де за допомогою послідовних операцій згортки (convolution) та пулінгу (pooling) виділяються високорівневі абстрактні ознаки. На виході формується числова матриця — ембедінг, що є цифровим зліпком обличчя.

4. Статистичний аналіз схожості (Verification/Classification).

Порівняння поточного дескриптора з еталонним масивом даних здійснюється шляхом розрахунку метрик відстані на основі функції втрат Triplet Loss або ArcFace Loss.

Важливою перевагою використання OpenFace та DeepFace в інженерній практиці є абсолютна прозорість вихідного коду (Open-Source ліцензії MIT/Apache), кросплатформність та безкоштовність розгортання. На еталонному світовому датасеті Labeled Faces in the Wild (LFW) дані фреймворки демонструють найвищу точність розпізнавання, яка перевищує 99.2%. Крім того, розробнику надається гнучкість у виборі математичного апарату порівняння — можливість самостійно перевизначати порогові значення

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		21

(thresholds) для евклідової відстані, квадратів різниць або косинусної подібності [11, 13].

Проте, в контексті специфіки нашої задачі проектування автономної системи автентифікації, дані рішення мають суттєві архітектурні недоліки.

Комплексні ансамблеві моделі глибокого навчання складаються з десятків мільйонів параметрів (ваг), що вимагає виконання колосального обсягу операцій з рухомою комою (FLOPS) під час прямого проходу (forward pass).

Для забезпечення стабільної частоти обробки відеопотоку у реальному часі (понад 25 FPS) ці фреймворки вимагають обов'язкового використання потужних графічних прискорювачів (GPU) з підтримкою архітектури паралельних обчислень NVIDIA CUDA. При спробі локального виконання коду виключно на центральному процесорі (CPU) звичайного офісного комп'ютера або середньостатистичного ноутбука, затримка обробки одного кадру може сягати від 200 до 800 мс. Це призводить до падіння швидкодії до 2–5 FPS, що є критично недопустимим для динамічних систем контролю access-менеджменту та суттєво погіршує якість користувацького досвіду (UX) [14].

Таким чином, проведений аналіз систем OpenFace та DeepFace остаточно підтверджує науково-технічну гіпотезу моєї дипломної роботи. Для створення швидкої та автономної системи біометричного захисту, яка здатна функціонувати на низькопродуктивних CPU-пристроях у реальному часі, недоцільно використовувати «важкі» неймережеві класифікатори face recognition. Замість цього необхідно спроектувати гібридний софтверний стек: використовувати легку спеціалізовану модель (наприклад, конвеєр MediaPipe Face Mesh) виключно для миттєвого виділення просторової топології 3D-координат обличчя, а фінальну логіку автентифікації реалізувати на базі математичного апарату класичної лінійної алгебри та векторизованих антропометричних пропорцій. Це дозволить зберегти високу точність розпізнавання при мінімальних витратах обчислювальних ресурсів.

					ІАЛЦ.467200.003 ПЗ	Арк.
						22
Зм.	Арк.	№ докум.	Підпис	Дата		

1.4 Порівняльний аналіз програмних аналогів та постановка задачі

Проведений у попередніх підрозділах огляд існуючих методів та програмних засобів біометричної автентифікації дозволяє здійснити їх об'єктивне порівняння. Для вибору оптимального шляху проектування власної інтелектуальної системи необхідно систематизувати переваги та недоліки ключових систем-аналогів: апаратно-програмного комплексу Face ID, комерційного SDK Luxand та відкритих наукових фреймворків OpenFace/DeerFace [7].

Результати порівняльного аналізу наведено у таблиці 1.1.

Таблиця 1.1. — Переваги та недоліки існуючих систем біометричної автентифікації обличчя

Система-аналог	Використовувані технології	Переваги	Недоліки
Face ID (Apple)	3D-антропометрика, інфрачервона проекція (TrueDepth), Neural Engine	1. Безпрецедентно високий рівень безпеки (FAR 1:1 000 000) 2. Вбудований апаратний захист від атак підміни (spoofing) та liveness detection	1. Закрита пропрієтарна екосистема (тільки iOS) 2. Жорстка прив'язка до дороговартісного специфічного обладнання (IR-сенсори)

Продовження таблиці 1.1

Система-аналог	Використовувані технології	Переваги	Недоліки
Face ID (Apple)	3D-антропометрика, інфрачервона проекція (TrueDepth), Neural Engine	3. Безпрецедентно високий рівень безпеки (FAR 1:1 000 000) 4. Вбудований апаратний захист від атак підміни (spoofing) та liveness detection 5. Незалежність від умов зовнішнього освітлення	3. Закрита пропрієтарна екосистема (тільки iOS) 4. Жорстка прив'язка до дороговартісного специфічного обладнання (IR-сенсори) 5. Неможливість портування на стандартні персональні комп'ютери
Luxand FaceSDK	Статистичні методи та нейронні мережі (екстракція 70+ точок)	1. Висока швидкодія на звичайних центральних процесорах (CPU) 2. Стійкість до часткових оклюзій (маски, окуляри) та поворотів голови до 35° 3. Підтримка веб-камер низької роздільної здатності	1. Закритий вихідний код (архітектура "чорної скриньки") 2. Неможливість налаштування внутрішнього математичного апарату мережі 3. Висока вартість комерційної ліцензії

Кінець таблиці 1.1

Система-аналог	Використовувані технології	Переваги	Недоліки
OpenFace / DeepFace	Глибокі згорткові нейронні мережі (CNN: FaceNet, VGG-Face)	1. Абсолютна прозорість вихідного коду (Open Source) та безкоштовність 2. Найвища точність на еталонних наборах даних (понад 99% на LFW) 3. Гнучкість у виборі математичної метрики порівняння (евклідова відстань, косинусна подібність)	1. Надвисока ресурсомісткість ансамблевих моделей 2. Критична залежність від наявності потужного графічного прискорювача (GPU) для роботи в реальному часі (25+ FPS) 3. Значні затримки та зниження FPS при виконанні на CPU звичайного офісного комп'ютера

Проведений аналіз наочно демонструє наявність суттєвого технологічного розриву на сучасному ринку програмного забезпечення між надточними, але ресурсомісткими академічними моделями та швидкими, проте закритими комерційними системами. Для локальних систем контролю та управління доступом на базі звичайних персональних комп'ютерів використання важких згорткових фреймворків виявляється технічно недоцільним через брак обчислювальних ресурсів, а впровадження пропрієтарних апаратних засобів є економічно невиправданим.

З метою точнішої детермінації вимог до розробки та наочного обґрунтування наукової новизни бакалаврського проєкту, було виконано декомпозицію ключових технічних та експлуатаційних параметрів існуючих

рішень у порівнянні із проєктувальною софтверною системою. Результати цього порівняння зафіксовано у формі бінарної матриці відповідності критеріям, яка наведена у таблиці 1.2.

Таблиця 1.2 — Матриця відповідності технічних та експлуатаційних критеріїв порівнюваних систем автентифікації

Критерії порівняння та оцінки систем біометрії	Face ID (Apple)	Luxand FaceSDK	OpenFace / DeepFace	Дипломна система
Незалежність від спеціалізованого апаратного забезпечення (IR-датчиків)	—	+	+	+
Відсутність жорсткої залежності від потужних графічних прискорювачів (GPU)	—	+	—	+
Забезпечення високої частоти кадрів (30+ FPS) суто на обчисленнях CPU	—	+	—	+
Автономність та локальність обробки біометричних даних (Privacy-first)	+	+	+	+
Висока щільність антропометричного моделювання (понад 400 реперних точок)	+	—	—	+
Відкритість вихідного коду застосунку та відсутність ліцензійних витрат	—	—	+	+

Систематизація даних, проведена у представлений матриці відповідності критеріям, дозволяє чітко сформулювати наукову новизну та практичну

цінність випускної кваліфікаційної роботи. Наукова новизна дослідження полягає у розробці та математичному обґрунтуванні гібридного, оптимізованого софтверного конвеєра біометричної автентифікації, який успішно усуває класичний компроміс між обчислювальною складністю та точністю розпізнавання.

На відміну від відомих підходів, де глибокі нейронні мережі використовуються на всіх етапах обробки, у запропонованій архітектурі легка регресійна нейромережа застосовується виключно на початковому етапі для миттєвої локалізації просторових координат. Фінальний аналіз, векторизація та порівняння структурних шаблонів обличчя повністю перенесені на математичний апарат векторизованих лінійних обчислень, що забезпечує радикальне зниження алгоритмічної складності та мінімізує кількість необхідних операцій з рухомою комою.

Практична цінність отриманих результатів полягає у створенні повністю автономного, кросплатформного програмного продукту, здатного функціонувати в реальному часі із частотою понад 30 кадрів на секунду на центральних процесорах пристроїв низької продуктивності, зокрема звичайних офісних ноутбуків чи персональних комп'ютерів.

Завдяки повній відмові від обчислень на GPU та інтеграції моделі Face Mesh на 468 ключових точок, система розгортається без залучення дороговартісного обладнання, використовуючи стандартні інтегровані веб-камери. Реалізована концепція локального збереження даних у безсерверній СУБД SQLite та бінарна серіалізація шаблонів гарантують найвищий рівень конфіденційності користувачів за принципом суворої ізоляції біометричних дескрипторів без їх передачі у відкриті телекомунікаційні мережі Інтернет.

З огляду на це, в межах даного дипломного проєкту постає актуальна науково-практична задача, яка полягає у проєктуванні та програмній реалізації зазначеної інтелектуальної локальної системи.

					ІАЛЦ.467200.003 ПЗ	Арк.
						27
Зм.	Арк.	№ докум.	Підпис	Дата		

Для досягнення визначеної мети необхідно послідовно вирішити комплекс взаємопов'язаних інженерних завдань, що включають вибір раціональної алгоритмічної бази комп'ютерного зору, проектування високоефективного обчислювального ядра математичного порівняння, побудову захищеної підсистеми локального зберігання шаблонів на основі криптографічного ґешування, розробку ергономічного багатопотокового графічного інтерфейсу користувача мовою Python та проведення всебічних експериментальних досліджень для валідації точності системи за критеріями помилкового доступу та помилкової відмови у різних експлуатаційних умовах.

Вирішення цих завдань дозволить отримати оптимізоване, незалежне від сторонніх хмарних рішень та ліцензійних обмежень програмне забезпечення, що повною мірою відповідає сучасним вимогам до софтверних систем інформаційної безпеки.

					ІАЛЦ.467200.003 ПЗ	Арк.
						28
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВОК ДО РОЗДІЛУ 1

У першому розділі дипломного проєкту було здійснено комплексний теоретичний аналіз предметної області біометричної автентифікації та детально досліджено існуючі методи розпізнавання особи за геометрією обличчя. Виявлено, що ключові складнощі обробки та аналізу растрових зображень у реальному часі пов'язані з високою розмірністю простору ознак, нелінійністю вхідних інформаційних сигналів, а також стохастичним характером деструктивних чинників зовнішнього середовища, таких як коливання рівня освітленості, зміна просторових ракурсів голови (повороти у трьох площинах) та наявність часткових оклюзій.

Класичні методи комп'ютерного зору (зокрема, каскадні класифікатори Хаара або статистичні дискримінанти LBP) забезпечують високу швидкість обчислень на центральних процесорах, проте демонструють низьку стійкість до зміни зовнішніх умов і непридатні для точного антропометричного аналізу. З іншого боку, передові глибокі згорткові нейронні мережі (CNN) гарантують надвисоку точність розпізнавання, але є надмірно ресурсомісткими, що вимагає використання потужних графічних прискорювачів (GPU) та унеможлиблює їх плавне функціонування на стандартних локальних ЕОМ. Побудова сучасних локальних засобів захисту потребує впровадження гібридних підходів, які б поєднували аналітичну швидкість геометричного моделювання зі стійкістю легких регресійних нейромережевих детекторів ключових точок.

Окрім цього, було проведено детальне порівняльне дослідження існуючих програмних та апаратно-програмних рішень, що представлені на ринку інформаційних технологій. Розглянуто архітектурні принципи функціонування апаратного комплексу Face ID від компанії Apple, комерційного пакета Luxand FaceSDK, а також відкритих фреймворків OpenFace та DeepFace. За результатами аналізу було сформовано порівняльну матрицю переваг та

					ІАЛЦ.467200.003 ПЗ	Арк.
						29
Зм.	Арк.	№ докум.	Підпис	Дата		

недоліків кожного продукту. Встановлено, що пропрієтарні системи є закритими та жорстко прив'язаними до дороговартісного заліза, комерційні SDK мають високу вартість ліцензування та архітектуру «чорної скриньки», а відкриті неймережеві бібліотеки викликають критичне падіння частоти кадрів (FPS) при запуску на CPU.

На основі виявленого технологічного розриву було обґрунтовано й сформульовано постановку задачі на проектування власної інтелектуальної системи біометричної автентифікації. Об'єктом розробки визначено автономний локальний десктопний застосунок мовою Python, що функціонує без передачі біометричних профілів у зовнішнє мережеве середовище. Для вирішення оптимізаційних задач швидкодії та точності було прийнято інженерне рішення реалізувати обчислювальне ядро системи на базі технології Landmark Detection високої щільності, а саме моделі Face Mesh, яка екстрагує декартові координати сітки з 468 ключових точок обличчя користувача. Математичний апарат порівняння сформованих антропометричних векторів ознак (ембеддінгів) базуватиметься на розрахунку просторових метрик близькості — евклідової відстані та косинусної подібності, а зберігання шаблонів відбуватиметься в ізольованій базі даних SQLite. Головною метою подальшої розробки є створення оптимізованого програмного комплексу та експериментальне дослідження його ефективності за критеріями обчислювальної швидкодії й точності на основі оцінки коефіцієнтів FAR та FRR.

					ІАЛЦ.467200.003 ПЗ	Арк.
						30
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ОБҐРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ТА МАТЕМАТИЧНИЙ АПАРАТ

2.1 Детальний огляд алгоритмів, які лягли в основу об'єкта розробки

Проектування інтелектуальної системи біометричної автентифікації в реальному часі вимагає побудови чіткого алгоритмічного конвеєра (pipeline) обробки відеопотоку. Обчислювальне ядро розроблюваного програмного забезпечення має послідовно вирішувати задачі дискретизації відеокадрів, детекції просторового розташування об'єкта, його геометричного вирівнювання, фільтрації шумів, надточного виділення ключових ознак та їх фінального метричного порівняння з еталоном.

Для забезпечення стабільної роботи застосунку на центральному процесорі (CPU) без залучення спеціалізованих графічних прискорювачів (GPU), архітектуру системи побудовано на базі сучасних полегшених регресійних моделей комп'ютерного зору та класичних методів лінійної алгебри. Це дозволяє оптимізувати використання оперативної пам'яті та мінімізувати часову складність обчислень [4, 8].

2.1.1 Огляд алгоритмів Landmark Detection для виділення 468 ключових точок

Основою підсистеми екстракції біометричних ознак у даному проєкті є технологія Landmark Detection (детекція ключових точок обличчя) високої щільності, реалізована за допомогою фреймворку MediaPipe Face Mesh. Алгоритм працює за принципом двоетапного нейромережевого конвеєра.

					ІАЛЦ.467200.003 ПЗ	Арк.
						31
Зм.	Арк.	№ докум.	Підпис	Дата		

Спершу блок первинної локалізації (BlazeFace) — полегшена згорткова нейромережа, яка сканує весь вхідний відеокадр, визначає наявність обличчя та повертає його обмежувальний прямокутник (bounding box) разом із 6 базовими підхопиточними точками (очі, вуха, ніс, рот). Це дозволяє відсікти зайвий фон та звузити регіон інтересу (ROI).

Потім блок регресії сітки (Face Mesh) — глибока нейромережа, яка працює виключно всередині виділеного ROI. Вона не просто класифікує текстуру, а виконує прямий регресійний аналіз, прогножуючи декартові координати для 468 ключових антропометричних точок у тривимірному просторі [12, 21].

Математична особливість моделі полягає в тому, що кожна із 468 точок описується вектором координатного простору:

$$P_i = (x_i, y_i, z_i), \quad i \in [1, 468]$$

де x_i та y_i — нормалізовані координати точки на площині кадру (в інтервалі $[0.0, 1.0]$ відносно ширини та висоти зображення);

z_i — відносна глибина (depth), яка описує віддаленість точки від умовної центральної площини обличчя. Координата z розраховується нейромережею на основі аналізу перспективних спотворень дрібних елементів текстури шкіри та має той самий масштаб, що й координата x .

Завдяки такій високій щільності (468 точок), топологічна схема розподілу яких наочно проілюстрована на рисунку 2.1, координатна сітка повністю покриває всі ключові анатомічні зони об'єкта розпізнавання: контури очей (по 32 точки на кожне), брови (по 23 точки), контури губ (80 точок) та загальний овал обличчя.

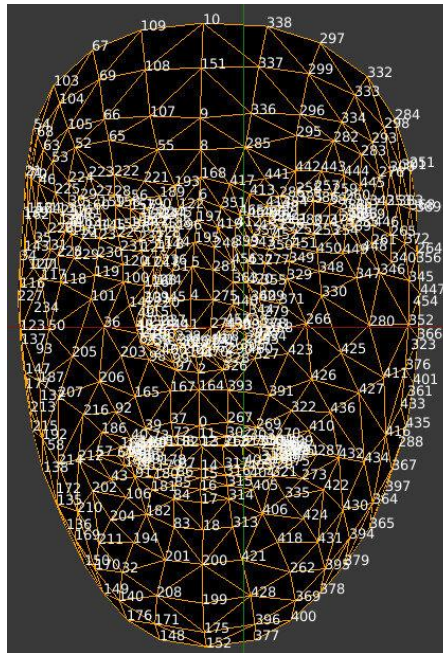


Рисунок 2.1 – Топологічна схема розподілу 468 антропометричних точок алгоритму Face Mesh

Висока щільність точок забезпечує надлишковість та стабільність біометричного опису: навіть при зміні міміки (посмішка, відкриття рота) відносне розташування жорстких кісткових зон (наприклад, відстань між внутрішніми куточками очей або переніссям) залишається незмінним, що дозволяє повністю нівелювати вплив виразу обличчя на процедуру автентифікації [21].

2.1.2 Математична модель розрахунку векторів ознак (евклідова відстань та косинусна подібність)

Пряме використання сирих координат 468 точок ($468 \times 3 = 1404$ значення) як біометричного шаблону є неефективним, оскільки вони змінюються при будь-якому наближенні користувача до камери або зміщенні в межах кадру. Для побудови інваріантного до масштабу та зсуву біометричного

ембедінгу (антропометричного вектора ознак) у системі реалізовано математичну модель розрахунку відносних просторових інтервалів [4].

З повної множини точок виділяється підмножина найбільш стабільних індексів, що описують фундаментальну архітектуру обличчя. Нехай $d(P_i, P_j)$ — лінійна евклідова відстань між точками P_i та P_j у тривимірному просторі, яка обчислюється за формулою:

$$d(P_i, P_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

Для досягнення повної масштабованої інваріантності (щоб відстань до камери не впливала на результат), обирається базовий анатомічний інтервал — нормалізуюча константа d_{base} . Найчастіше як d_{base} використовують відносно стабільну індивідуальну відстань між внутрішніми куточками очей особи користувача. Кожен елемент f_k фінального вектора ознак U розраховується як відношення конкретної геометричної відстані до базової:

$$f_k = \frac{d(P_i, P_j)}{d_{base}}$$

Сформований таким чином антропометричний вектор ознак $U = [f_1, f_2, \dots, f_M]^T \in \mathbb{R}^M$ (де M — кількість відібраних геометричних зв'язків) описує унікальні пропорції обличчя [16].

У процесі верифікації (порівняння «один-до-одного») система порівнює поточний вектор ознак U , отриманий з камери, з еталонним вектором $V = [f'_1, f'_2, \dots, f'_M]^T$, який було згенеровано під час реєстрації та вилучено з бази даних SQLite. Порівняння здійснюється за двома метриками:

- Евклідова відстань у просторі ознак (D_E):

Визначає геометричну близькість між двома точками в $\$M\$$ -вимірному просторі:

$$D_E(U, V) = \sqrt{\sum_{k=1}^M (f_k - f'_k)^2}$$

Рішення про успішну автентифікацію приймається за умовою $D_E(U, V) \leq \theta$, де θ — емпірично підібраний поріг чутливості (threshold). Чим ближче значення до нуля, тим вища ймовірність збігу осіб.

- Косинусна подібність (S_C):

Оцінює косинус кута між двома векторами, що дозволяє нівелювати можливі лінійні спотворення модулів векторів при різких нахилах голови:

$$S_C(U, V) = \frac{U \cdot V}{\|U\| \|V\|} = \frac{\sum_{k=1}^M f_k f'_k}{\sum_{k=1}^M f_k^2 \cdot \sqrt{\sum_{k=1}^M (f'_k)^2}}$$

Значення S_C лежить в інтервалі $[0,1]$ (для невід'ємних просторів ознак). Значення $S_C = 1$ свідчить про повну колінеарність векторів (абсолютний збіг пропорцій). Доступ надається, якщо $S_C(U, V) \geq \tau$, де τ — поріг допуску.

Дані інтерпретації можна переглянути на рисунку 2.2.

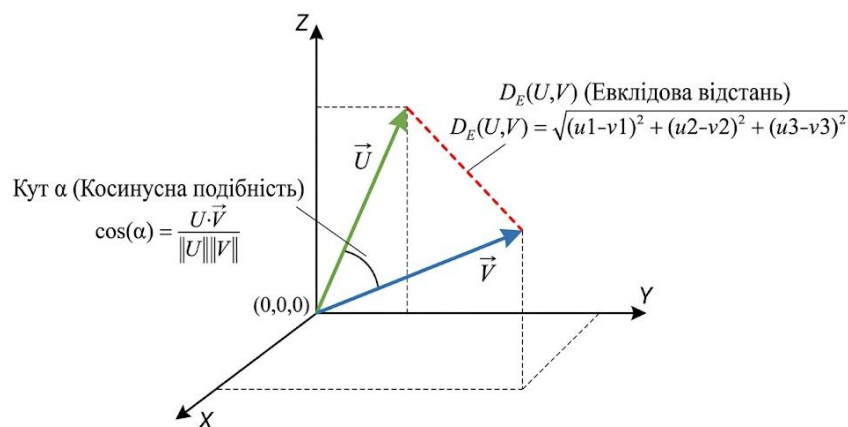


Рисунок 2.2 – Геометрична інтерпретація метрик Евклідової відстані та Косинусної подібності у багатовимірному просторі біометричних ознак

Наведена геометрична інтерпретація наочно демонструє відмінність між двома підходами до оцінки близькості біометричних дескрипторів. Це обґрунтовує необхідність їхнього сумісного використання в обчислювальному ядрі розроблюваної системи. Евклідова відстань відображає абсолютну лінійну дистанцію між векторами поточного кадру V_{curr} та еталона V_{ref} , тоді як косинусна подібність оцінює лише кутову девіацію θ між їхніми напрямками, ігноруючи абсолютні модулі.

Використання евклідової метрики є ефективним у стабільних умовах зйомки, оскільки суттєва зміна геометрії обличчя одразу збільшує відстань D_E . Однак, у реальних умовах, зміна нахилу голови або особливості об'єктива можуть викликати лінійні флуктуації координат точок. Навіть при нормалізації, це може штучно збільшити евклідову відстань і призвести до хибної відмови (зростання FRR).

Для компенсації цього ефекту інтегровано косинусну подібність. Якщо пропорції обличчя збігаються, але масштабувалися через зміну ракурсу, вектори V_{curr} та V_{ref} залишатимуться співнапрямленими, а косинус кута між ними наближатиметься до одиниці.

Комбінація цих двох метрик створює надійну двофакторну модель автентифікації. Евклідова відстань жорстко контролює лінійну відповідність, мінімізуючи ризик помилкового доступу (FAR). Водночас, косинусна подібність адаптує систему до динаміки відеопотоку, зменшуючи ймовірність відхилення легітимного користувача. Такий підхід, реалізований засобами класичної лінійної алгебри, гарантує високу точність розпізнавання при економному використанні ресурсів процесора, що є критичним для локальних систем контролю доступу.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		36

2.1.3 Методи попередньої обробки, нормалізації освітлення та вирівнювання обличчя

Сирі кадри, що надходять з веб-камери через модуль OpenCV, зазнають впливу випадкових шумів матриці та нерівномірного зовнішнього освітлення. Для зниження показників FRR (помилкових відхилень легітимного користувача) у програмному комплексі реалізовано обов'язковий етап попередньої підготовки зображення (Image Preprocessing).

1. Просторове вирівнювання (Face Alignment)

Нахил голови користувача вбік (кут roll) створює помилкову асиметрію координат на 2D-площині зображення. Для усунення цього ефекту перед екстракцією повної сітки виконується афінне перетворення кадру. За допомогою первинних базових точок очей обчислюється кут нахилу лінії очей відносно горизонтальної осі кадру:

$$\alpha = \arctan\left(\frac{y_{right_eye} - y_{left_eye}}{x_{right_eye} - x_{left_eye}}\right)$$

На основі розрахованого кута α та математичного центра обличчя $C(x_c, y_c)$ формується матриця афінного перетворення повороту M_{affine} . Кадр трансформується (повертається на кут $-\alpha$), що призводить обличчя до суворого вертикального орієнтування (анфас), мінімізуючи просторові похибки детекції.

2. Нормалізація освітлення

Для усунення жорстких тіней (наприклад, коли джерело світла розташоване лише з одного боку від користувача) застосовується алгоритм CLAHE (Contrast Limited Adaptive Histogram Equalization — адаптивне вирівнювання гістограми з обмеженням контрасту). На відміну від класичного глобального вирівнювання гістограми, яке може призвести до появи цифрових шумів та переосвітлених зон, CLAHE розбиває зображення на локальні блоки (тайли, зазвичай розміром 8×8 пікселів) [5, 17].

Для кожного блоку індивідуально обчислюється функція розподілу яскравості, яка вирівнює локальний контраст за формулою перерозподілу гістограми:

$$g = [g_{max} - g_{min}] \cdot P(f) + g_{min}$$

де $P(f)$ — інтегральна функція розподілу яскравості вхідного пікселя f , g_{min} , g_{max} — граничні значення вихідного динамічного діапазону.

Згідно з послідовністю конвеєра попередньої обробки відеопотоку, що проілюстрована на рисунку 2.3, для запобігання надмірному підсиленню цифрового шуму на локальних однорідних ділянках кадру, висота гістограми обрізається на заданому порозі (Clip Limit), а всі надлишкові пікселі рівномірно перерозподіляються за іншими рівнями яскравості.

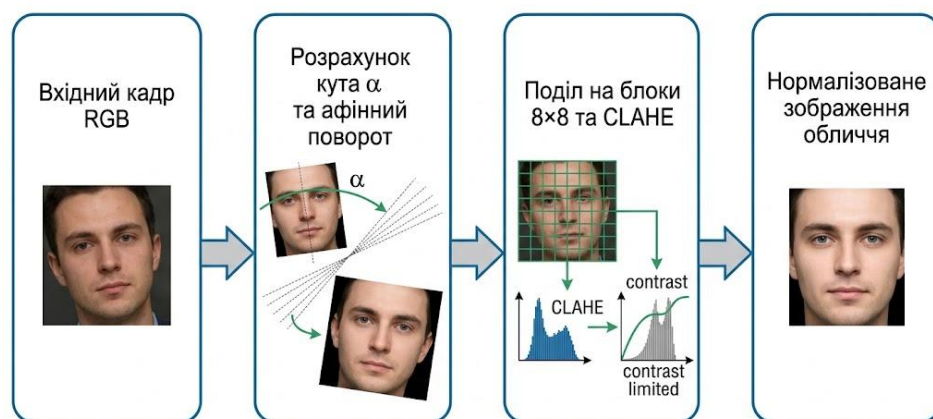


Рисунок 2.3 – Блок-схема етапів попередньої інженерної обробки та нормалізації вхідного відеокадру

На фінальному етапі цього процесу виконується двовимірна білінійна інтерполяція між просторовими межами сусідніх блоків сітки розміром 8×8 пікселів. Це дозволяє повністю ліквідувати видимі артефакти сегментації та різкі граничні перепади яскравості, що забезпечує високу чіткість і плавність текстури обличчя перед виконанням фінального антропометричного аналізу.

2.2 Мова програмування

Важливим етапом архітектурного проектування інтелектуальних систем є обґрунтований вибір базової мови програмування, яка визначає синтаксичну структуру вихідного коду, швидкість виконання обчислювальних процесів та ефективність керування апаратними ресурсами ЕОМ. При розробці локального десктопного застосунку для біометричної автентифікації на основі антропометрики обличчя висуваються жорсткі інженерні вимоги як до швидкості обробки поточкових даних комп'ютерного зору (CV), так і до регламенту проектування компонентів графічного інтерфейсу користувача (GUI) та логіки взаємодії з локальною базою даних.

2.2.1 Мова Python

Для практичної реалізації обчислювального ядра та інтерфейсної частини розроблюваної системи біометричного захисту доступу було обрано мову програмування Python. Вона підтримує мультипарадигмальний підхід, дозволяючи поєднувати об'єктно-орієнтоване, функціональне та процедурне програмування, що є критично важливим для гнучкої побудови архітектури сучасних систем контролю та управління доступом (СКУД) [20].

З інженерної точки зору, Python функціонує на базі віртуальної машини, яка послідовно виконує згенерований компілятором байт-код. Базова еталонна реалізація мови (інтерпретатор CPython) написана на мові C, що забезпечує низькорівневу сумісність із системними бібліотеками операційної системи та дозволяє безперешкодно інтегрувати високопродуктивні модулі, написані на компільованих мовах. Динамічна типізація та автоматичне керування пам'яттю за допомогою вбудованого механізму підрахунку посилань і збирача сміття (Garbage Collector) суттєво знижують імовірність виникнення критичних

					ІАЛЦ.467200.003 ПЗ	Арк.
						39
Зм.	Арк.	№ докум.	Підпис	Дата		

помилки сегментації оперативної пам'яті (RAM), значно прискорюючи загальний цикл тестування та валідації програмного забезпечення [6, 7].

2.2.2 Переваги мови Python для AI-задач та обробки даних

Домінування мови Python у сфері штучного інтелекту (AI) та комп'ютерного зору не є випадковим і зумовлене її унікальною екосистемою. Об'єктивним підтвердженням цього інженерного вибору є актуальні статистичні дані світових аналітичних індексів [6].

Для поточної інженерної задачі розпізнавання обличчя на основі щільних антропометричних сіток мова Python надає такі фундаментальні системні переваги:

1. Наявність низькорівневих оптимізованих обгорт (Native Wrappers)

Найбільш ресурсомісткі компоненти обчислювального ядра нашого застосунку, такі як бібліотека комп'ютерного зору OpenCV, математичний фреймворк NumPy та моделі екстракції точок сітки від MediaPipe, хоч і викликаються через високорівневий Python API, під капотом реалізовані та скомпільовані на мовах C/C++. Це означає, що матричні обчислення над кадрами та операції детекції виконуються безпосередньо у машинному коді з максимальною швидкістю, а Python слугує лише гнучким інженерним конвеєром для об'єднання цих модулів.

2. Ефективна робота з багатовимірними масивами даних

Бібліотека NumPy дозволяє виконувати складні векторизовані операції над масивами координат 468 ключових точок обличчя без використання повільних циклів на рівні інтерпретатора. Розрахунок евклідових відстаней та косинусної подібності для сотень просторових зв'язків одночасно делегується оптимізованим процесорним інструкціям (зокрема SIMD), що мінімізує часову складність алгоритму.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		40

3. Швидкість розгортання та інтеграції підсистем

Створення локальної реляційної бази даних та графічного інтерфейсу користувача за допомогою зв'язки SQLite + PyQt реалізується у межах єдиного програмного середовища з мінімальним обсягом шаблонного коду (boilerplate code), що дозволяє сфокусувати інженерні зусилля на оптимізації математичних моделей автентифікації особи.

2.2.3 Аналіз альтернатив (Java/C++) та їх недоліки для поточної задачі

У процесі архітектурного пошуку та обґрунтування технологічного стека було детально проаналізовано альтернативні мови програмування промислового рівня, зокрема компільовану мову C++ та кросплатформну мову Java, які традиційно застосовуються для розробки десктопного ПЗ.

Мова C++ є визнаним еталоном продуктивності для систем реального часу. Вона забезпечує пряму компіляцію вихідного тексту у бінарний машинний код та безпосередній контроль над виділенням і звільненням оперативної пам'яті.

Недоліки для поточної задачі — розробка комплексного десктопного застосунку з інтеграцією ШІ-моделей та складного GUI на мові C++ супроводжується надвисокою обчислювальною та синтаксичною складністю. Ручне керування пам'яттю підвищує ризики виникнення витоків даних (memory leaks) при безперервному виділенні пам'яті під гігабайти відеокадрів, що надходять з камери у реальному часі. Окрім того, відсутність стандартизованих інструментів для швидкого розгортання та конфігурації ШІ-інструментарію (на кшталт MediaPipe Face Mesh) суттєво збільшує час розробки (Development Time), що є критичним фактором в умовах обмеженого часового регламенту бакалаврського проєкту.

					ІАЛЦ.467200.003 ПЗ	Арк.
						41
Зм.	Арк.	№ докум.	Підпис	Дата		

Мова Java виконується всередині віртуального середовища JVM (Java Virtual Machine) та пропонує строгу типізацію й надійну об'єктно-орієнтовану структуру компонентів.

Недоліки для поточної задачі — для підключення низькорівневих бібліотек комп'ютерного зору в Java необхідно використовувати складний інтерфейс взаємодії з рідним кодом JNI (Java Native Interface). Це створює суттєві обчислювальні накладні витрати на маршалінг та копіювання масивів даних між керованою пам'яттю JVM та некерованою пам'яттю рідних бібліотек. Крім того, автоматичний збирач сміття (Garbage Collector) в JVM працює стохастично. Періодичні повні зупинки потоків виконання програми для очищення пам'яті (так звані «Stop-the-world» паузи) є неприпустимими при стабільній обробці відеопотоку, оскільки вони викликають раптові мікрозависання та падіння показника частоти кадрів (FPS), що критично для систем автентифікації [7, 8].

2.2.4 Підсумок щодо вибору мови локального середовища

Таким чином, на основі проведеного багатокритеріального інженерного аналізу встановлено, що мова Python є оптимальним компромісним рішенням для розробки локальної інтелектуальної системи автентифікації. Вона дозволяє абстрагуватися від низькорівневих проблем ручного керування пам'яттю, забезпечуючи високу швидкість написання, тестування та модифікації коду, але при цьому гарантує необхідну для СКУД швидкодію за рахунок використання рідних (native) модулів MediaPipe та OpenCV, скомпільованих на C/C++. Це дозволяє повністю виконати технічні вимоги щодо безперебійної обробки відеопотоку в реальному часі без перевантаження обчислювальних потужностей локальної ЕОМ користувача.

					ІАЛЦ.467200.003 ПЗ	Арк.
						42
Зм.	Арк.	№ докум.	Підпис	Дата		

2.3 Технології та бібліотеки

Ефективна реалізація математичного апарату біометричної автентифікації та обробки відеопотоку в реальному часі вимагає залучення надійних, перевірених часом та апаратно оптимізованих програмних бібліотек. У межах розробки даної автономної локальної системи було сформовано технологічний стек на базі екосистеми Python. Головною інженерною перевагою такого вибору є використання високорівневих абстракцій (API) для маніпуляції низькорівневими модулями, які під капотом скомпільовані мовами C/C++. Такий підхід гарантує необхідний баланс між максимальною обчислювальною продуктивністю ядра комп'ютерного зору та гнучкістю розробки графічного інтерфейсу клієнта.

2.3.1 OpenCV (обробка відеопотоку)

OpenCV (Open Source Computer Vision Library) — це фундаментальна бібліотека алгоритмів комп'ютерного зору, обробки растрових зображень та чисельних методів загального призначення. У розроблюваному програмному комплексі модуль `cv2` виступає не лише як інструмент захоплення відео, але й як повноцінна підсистема попередньої обробки (Image Preprocessing Pipeline) [22].

Безперервне зчитування кадрів із веб-камери пристрою у реальному часі здійснюється через клас `cv2.VideoCapture`. Використання внутрішніх буферів OpenCV дозволяє мінімізувати затримки (latency) між фізичним захопленням фотонів матрицею камери та передачею масиву пікселів у оперативну пам'ять програми [22].

За замовчуванням апаратні драйвери веб-камер та сам OpenCV повертають матрицю пікселів у форматі BGR (Blue, Green, Red). Проте сучасні

					ІАЛЦ.467200.003 ПЗ	Арк.
						43
Зм.	Арк.	№ докум.	Підпис	Дата		

нейромережеві моделі (включаючи MediaPipe) натреновані на стандартному просторі RGB. Використання оптимізованої функції `cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)` дозволяє виконувати конвертацію колірних каналів на рівні векторних інструкцій процесора практично миттєво.

Представлена структурно-технологічна схема на рисунку 2.4 відображає послідовність підготовки кадрів до антропометричного аналізу. Для нівелювання нерівномірного освітлення та жорстких тіней, які критично погіршують метрику FRR, у конвеєр інтегровано алгоритм локального вирівнювання гістограми CLAHE.

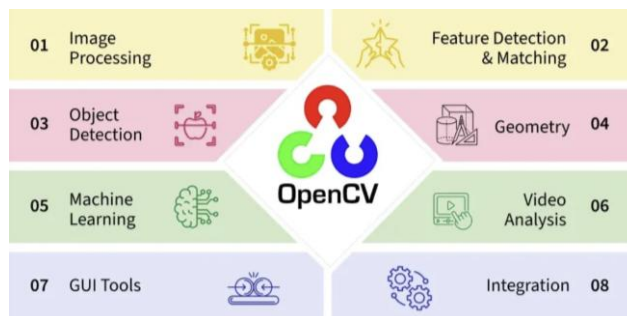


Рисунок 2.4 — Структурно-технологічна схема конвеєра попередньої обробки та візуалізації відеопотоку засобами бібліотеки OpenCV

Метод обробляє зображення окремими тайлами розміром 8x8 пікселів, де параметр `Clip Limit` обмежує надмірне зростання локального контрасту, запобігаючи появі пересвітів та підкреслюючи рельєф обличчя. Після цього за допомогою функцій `cv2.getRotationMatrix2D` та `cv2.warpAffine` виконується геометричне вирівнювання нахилу голови (Face Alignment). Розрахована за координатами очей матриця афінних перетворень розгортає обличчя у суворо фронтальну площину, що забезпечує просторову інваріантність лінійних розрахунків [17].

Окрім аналітичних обчислень, OpenCV реалізує рендеринг інтерфейсу для зворотного зв'язку з користувачем. Високорівневі методи `cv2.rectangle`,

cv2.line та *cv2.putText* накладають обмежувальні рамки детекції, топологічну сітку маркерів та текстові сповіщення безпосередньо на вихідну матрицю пікселів. Завдяки низькорівневій оптимізації ядра бібліотеки на мові C++, весь конвеєр попередньої обробки працює в єдиному обчислювальному потоці з графічним віджетом PyQt, що виключає затримки та гарантує плавну візуалізацію із частотою понад 30 кадрів на секунду на звичайному центральному процесорі.

2.3.2 MediaPipe (модель Face Mesh)

MediaPipe — це високорівневий кросплатформний фреймворк від компанії Google, призначений для проєктування та розгортання складних мультимодальних конвеєрів прикладного машинного навчання та комп'ютерного зору. Для вирішення ключової задачі даного дипломного проєкту — високоточної локалізації просторових антропометричних маркерів обличчя у реальному часі — в архітектуру обчислювального ядра було інтегровано спеціалізований модуль MediaPipe Face Mesh.

На відміну від класичних важких та монолітних згорткових нейромережових архітектур, які потребують значних апаратних ресурсів для розрахунку прямого проходу, цей фреймворк реалізує гнучкий графовий підхід до організації обчислень. Усі інформаційні потоки всередині системи описуються у вигляді спрямованого ациклічного графа калькуляторів (Calculator Graph), що дозволяє оптимізувати передачу тензорів на рівні низькорівневого рантайму TensorFlow Lite [12].

Процес екстракції геометричних ознак у модулі Face Mesh функціонує за концепцією каскадного двокомпонентного конвеєра, який послідовно поєднує етапи глобальної детекції та локальної регресії реперних маркерів. Така софтверна організація дозволяє уникнути надлишкового сканування всього

					ІАЛЦ.467200.003 ПЗ	Арк.
						45
Зм.	Арк.	№ докум.	Підпис	Дата		

растрового поля кожного кадру і радикально знижує загальну алгоритмічну складність системи під час обробки динамічного відеопотоку.

Структурно-логічну схему зазначеного обчислювального конвеєра неймережевої локалізації та принципи взаємодії його внутрішніх шарів наведено нижче на рисунку 2.5.

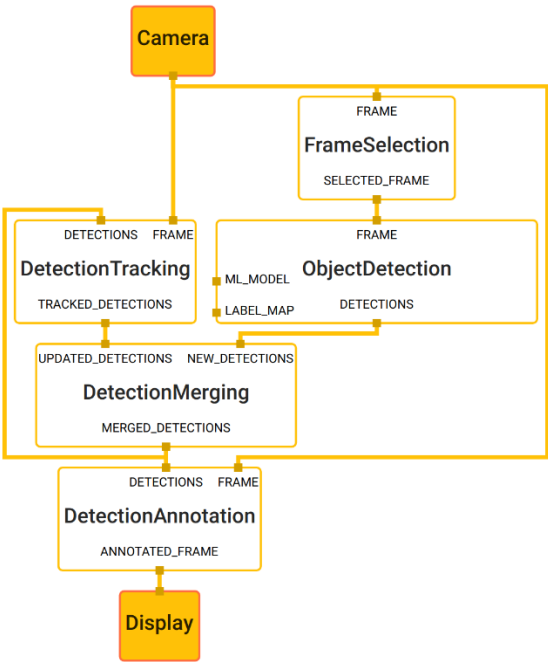


Рисунок 2.5 — Структурно-логічка схема двокомпонентного обчислювального конвеєра неймережевої локалізації MediaPipe Face Mesh

Детальний аналіз представленої архітектурної схеми дозволяє покроково декомпонувати математичні та програмні процеси, що протікають всередині кожного обчислювального вузла графа.

Першим базовим рівнем конвеєра є надвишвидка неймережева модель детекції BlazeFace, яка відповідає за первинний аналіз вхідного кадру, попередньо конвертованого бібліотекою OpenCV у колірний простір RGB. Модель BlazeFace функціонує із суттєво зменшеною роздільною здатністю зображення, що мінімізує кількість аналізованих ознак, і прогнозує просторові межі обличчя у вигляді обмежувального прямокутника. Цей детектор

спеціально оптимізований під низькопродуктивні десктопні центральні процесори за рахунок впровадження модифікованих легких блоків MobileNet, які реалізують глибинно-роздільні згортки та специфічні шари субдискретизації для прискорення агрегації просторових ознак без втрати інформації про локальні реперні зони.

Після того, як BlazeFace ізолює регіон інтересу (Region of Interest) та передає його координати далі по графу обчислень, активується другий, більш глибокий шар системи — регресійна нейромережева модель відновлення топології поверхні обличчя. Ця модель фокусується виключно на вирізаному фрагменті кадру і здійснює поточкове прогнозування тривимірних 3D-координат для 468 ключових точок.

Важливою математичною особливістю функціонування цього регресора є повернення нормалізованих відносних координат по осях абсцис та ординат у фіксованому діапазоні від 0.0 до 1.0. Такий підхід забезпечує повну математичну інваріантність алгоритму до фізичних параметрів кадру, роблячи роботу системи незалежною від роздільної здатності веб-камери пристрою користувача (як для застарілого формату 640x480 пікселів, так і для сучасного Full HD 1920x1080 пікселів), що повністю нівелює необхідність додаткового програмного масштабування або інтерполяції матриць пікселів.

Головною інженерною перевагою та чинником забезпечення високої швидкодії модуля MediaPipe є інтегрований усередину графа інтелектуальний механізм динамічного трекінгу (Tracking). Алгоритм не запускає ресурсомістку процедуру повної детекції всього кадру за допомогою BlazeFace на кожному дискретному кроці часу. Якщо обличчя користувача було успішно локалізоване на поточному кадрі з індексом N , то на наступному кадрі $N + 1$ регресійна сітка маркерів лише коригується у вузькому вікні пошуку на основі розрахунку векторів оптичного потоку та порівняння просторової послідовності.

Повне перезапускання первинного детектора BlazeFace ініціюється операційною системою конвеєра лише у випадку критичного зсуву об'єкта, його

					ІАЛЦ.467200.003 ПЗ	Арк.
						47
Зм.	Арк.	№ докум.	Підпис	Дата		

виходу за межі видимості веб-камери або різкої зміни ракурсу, що фіксується падінням внутрішньої метрики достовірності. Такий збалансований конвеєрний механізм дозволяє досягти стабільної частоти обробки відеопотоку на рівні понад 30 кадрів на секунду виключно на стандартних CPU-обчисленнях, що підтверджує раціональність обраного інженерного шляху [12, 21].

2.3.3 NumPy (матричні обчислення)

NumPy (Numerical Python) — це фундаментальна системна бібліотека для наукових та інженерних обчислень, яка лежить в основі фактично всього сучасного стека машинного навчання та комп'ютерного зору на мові Python. У розроблюваному обчислювальному ядрі локальної системи автентифікації ця бібліотека відіграє роль головного математичного рушія. Її основним завданням є агрегація, багатовимірна трансформація та статистичне порівняння біометричних векторів, отриманих після етапу нейромережевої екстракції ознак.

Хоча мова Python є надзвичайно гнучкою та зручною для прототипування високорівневої логіки, її застосування для прямих математичних розрахунків має серйозні архітектурні обмеження. Послідовна ітерація через тисячі просторових координат за допомогою стандартних циклів `for` є критично повільною через наявність глобального блокування інтерпретатора (Global Interpreter Lock, GIL) та накладні витрати на динамічну типізацію об'єктів під час виконання (runtime). NumPy ефективно вирішує цю фундаментальну проблему продуктивності завдяки використанню спеціалізованої структури даних `ndarray` (n-dimensional array). Ця структура являє собою суцільний, неперервний блок оперативної пам'яті (C-contiguous), де всі елементи мають суворий статичний тип (наприклад, `float32` або `float64`), що забезпечує максимальну дружність до кешу центрального процесора.

					ІАЛЦ.467200.003 ПЗ	Арк.
						48
Зм.	Арк.	№ докум.	Підпис	Дата		

У контексті спроектованого конвеєра обробки даних, першочерговою функцією NumPy є перетворення сирих, неструктурованих списків об'єктів, які повертає фреймворк MediaPipe, у нормалізовані двовимірні матриці. Вихідний масив із 468 просторових точок миттєво трансформується у матрицю розмірності 468×3 , де стовпці відповідають координатам X, Y та Z. Таке суворе матричне представлення є абсолютно необхідним для підготовки даних до подальшої швидкої математичної обробки та розрахунку відносних антропометричних інтервалів.

Наступним критично важливим аспектом використання NumPy є концепція векторизації обчислень. Замість того, щоб обчислювати відстані між реперними точками обличчя покроково, операції додавання, віднімання або піднесення багатовимірних масивів до квадрата виконуються одночасно для всієї множини даних без жодного циклу на рівні Python. Це досягається за рахунок безпосереднього виклику скомпільованого машинного коду на мові C/C++, який автоматично трансліює математичні матричні операції у низькорівневі апаратні векторні інструкції процесора (SIMD — Single Instruction, Multiple Data). Такий підхід дозволяє процесору виконувати кілька операцій із рухомою комою за один машинний такт, що радикально підвищує загальну пропускну здатність системи.

Фінальним та найвідповідальнішим етапом є застосування модуля NumPy для реалізації метрик біометричного порівняння. Обчислення евклідової відстані між еталонним ембедінгом та поточним вектором кадру виконується за допомогою єдиної оптимізованої функції *np.linalg.norm(u - v)*. Аналогічно, математичний апарат косинусної подібності реалізується через оптимізований скалярний добуток *np.dot* у поєднанні з діленням на модулі векторів. Завдяки перенесенню цієї важкої лінійної алгебри на векторизований рівень NumPy, загальний час верифікації одного кадру та розрахунку мір просторової близькості скорочується з кількох мілісекунд до десятків мікросекунд.

					ІАЛЦ.467200.003 ПЗ	Арк.
						49
Зм.	Арк.	№ докум.	Підпис	Дата		

2.3.4 PySide6 (для графічного інтерфейсу)

PySide6 (офіційний проєкт Qt for Python) — це сучасний та високоефективний набір прив'язок мови Python для кросплатформного інструментарію розробки графічних інтерфейсів користувача фреймворку Qt 6. У межах проєктування інтелектуальної системи біометричної автентифікації ця бібліотека виступає як базовий каркас для побудови ергономічного та високопродуктивного клієнтського середовища.

Вона забезпечує повну функціональність та взаємодію користувача з програмним комплексом через систему спеціалізованих вікон. Це включає модулі первинної реєстрації біометричного еталона, інтерактивні віджети для трансляції реального відеопотоку з накладеною антропометричною сіткою, а також адміністративні панелі керування доступом і налаштування порогів чутливості. Вибір саме PySide6 обґрунтований його офіційною корпоративною підтримкою, широким набором готових оптимізованих віджетів та низькою латентністю рендерингу. Оскільки ядро Qt 6 компілюється на низькому рівні мовою C++, це надає Python-застосунку нативну продуктивність візуалізації [24].

Вирішальним фактором вибору PySide6 є його розвинений архітектурний апарат для роботи з багатопотоковістю та асинхронними подіями. Розробка систем комп'ютерного зору в реальному часі вимагає вирішення критичної проблеми блокування інтерфейсу, відомої у системній інженерії як GUI Freeze. Оскільки графічна підсистема Qt базується на архітектурі єдиного головного потоку виконання (Main Thread), який безперервно обробляє чергу системних подій (Event Loop) та оновлює стан вікон, будь-яке тривале лінійне обчислення всередині цього потоку миттєво призводить до зупинення реакції інтерфейсу.

Якби процеси пооб'єктного захоплення кадрів з веб-камери, адаптивного CLAHE-вирівнювання гістограми, афінних перетворень для компенсації нахилу голови (Face Alignment) та подальшого прогону через нейромережевий

					ІАЛЦ.467200.003 ПЗ	Арк.
						50
Зм.	Арк.	№ докум.	Підпис	Дата		

детектор MediaPipe Face Mesh разом із розрахунком евклідової відстані NumPy виконувалися в межах Main Thread, застосунок повністю зависав би на час обчислень кожного кадру, втрачаючи здатність реагувати на дії користувача.

Для подолання цього фундаментального обмеження в архітектурі розроблюваної системи було реалізовано класичний софтверний паттерн «Виробник-Споживач» (Producer-Consumer), ізольований на рівні операційної системи за допомогою засобів класу QThread. Взаємодія між різними контекстами виконання побудована на базі асинхронного механізму сигналів та слотів за допомогою об'єктів Signal (з модуля PySide6.QtCore), що гарантує сувору потокобезпеку (Thread Safety) без ризику виникнення станів перегонів (race conditions) за ресурси оперативної пам'яті.

У спроектованій схемі створюється окремий фоновий робочий потік (Worker Thread), який бере на себе всю важку обчислювальну роботу і виступає в ролі «виробника» біометричних даних. Цей потік у безкінечному циклі захоплює кадри, конвертує формати, виділяє топологічну хмару з 468 точок, розраховує нормалізовані антропометричні вектори ознак і порівнює їх із базою даних SQLite. Після завершення обчислювального такту потік не модифікує інтерфейс напряду, а лише емітує асинхронний сигнал, безпечно передаючи сформоване зображення у форматі QImage разом із результатами двофакторної верифікації (булевым прапором та значеннями метрик) [18, 24].

Головний потік програми (UI Thread) у цьому конвеєрі відіграє виключно роль «споживача», який підписаний на відповідні слоти міжпотокової взаємодії. Отримавши асинхронне сповіщення про надходження нового обробленого біометричного дескриптора, Main Thread миттєво виконує перемальовування графічних компонентів, відображаючи поточне відео на графічному контейнері QLabel та оновлюючи колірні індикатори допуску. Завдяки такій глибокій декомпозиції процесів, графічна оболонка застосунку залишається абсолютно чуйною та зберігає плавність анімацій на еталонному рівні для користувацького інтерфейсу, незалежно від миттєвого

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		51

обчислювального навантаження або затримок у ядрі математичних алгоритмів. Це дозволяє забезпечити бездоганний користувацький досвід та високу стабільність функціонування системи контролю доступу в цілому [18, 24].

2.3.5 SQLite (для локальної бази даних еталонів)

SQLite — це високонадійна, вбудована реляційна система управління базами даних, бібліотека якої інтегрована безпосередньо у ядро Python (*sqlite3*). Вона реалізує повноцінний механізм SQL та підтримує вимоги транзакційності ACID (Атомарність, Узгодженість, Ізольованість, Довговічність).

Доцільність використання SQLite замість звичайних текстових файлів або клієнт-серверних баз даних (MySQL/PostgreSQL) обґрунтовується специфікою СКУД [25].

SQLite не вимагає інсталяції та налаштування окремого процесу-сервера. Уся база біометричних профілів, метадані користувачів та логі автентифікації зберігаються в одному зашифрованому локальному файлі на жорсткому диску пристрою, що критично знижує поверхню атаки для злоумисників.

Використання SQL-індексів дозволяє миттєво знаходити запис конкретного користувача (за його ID) серед тисяч еталонів.

Архітектура бази містить поле типу BLOB (Binary Large Object). Оскільки збереження фотографій облич є порушенням приватності та ризиком витоку, у поле BLOB записується виключно числовий вектор ознак (ембеддінг) користувача, з якого неможливо реконструювати оригінальне зображення [19].

2.3.6 Pickle та Hashlib (засоби серіалізації та безпеки даних)

Оскільки обрана для проєктування підсистеми зберігання реляційна база даних SQLite концептуально орієнтована на роботу зі стандартними

скалярними типами (такими як цілі числа або текстові рядки), виникає нетривіальна інженерна проблема інтеграції складних структур.

Антропометричні шаблони, що генеруються обчислювальним ядром, являють собою багатовимірні масиви NumPy, які неможливо напряму експортувати у таблицю без порушення їхньої розмірності та структури.

Для ефективного вирішення цієї задачі на стику бази даних та обчислювального ядра технологічний стек було доповнено стандартними модулями Python: `pickle` для трансформації структур та `hashlib` для забезпечення криптографічної конфіденційності.

Модуль `pickle` відіграє роль основного інструмента для глибокої бінарної серіалізації об'єктів. Перед безпосереднім записом еталонного профілю користувача в базу даних, сформований масив NumPy обробляється функцією `pickle.dumps()`.

Цей алгоритм повністю інкапсулює внутрішній стан масиву та перетворює його на безперервний потік байтів (byte-stream), який ідеально підходить для збереження у форматі BLOB (Binary Large Object) всередині SQLite.

Під час процесу автентифікації та зчитування еталону з диска застосовується симетричний зворотний процес десеріалізації за допомогою методу `pickle.loads()`. Це дозволяє відновити повноцінний математичний об'єкт у оперативній пам'яті комп'ютера за доли мілісекунд, суворо зберігаючи початкові типи даних елементів масиву (наприклад, `float32`) та забезпечуючи їхню миттєву готовність до векторизованих розрахунків.

Зі свого боку, бібліотека `hashlib` реалізує фундаментальні механізми захисту інфраструктури розробленого застосунку на рівні збереження облікових даних.

Відповідно до сучасних стандартів кібербезпеки та архітектурної парадигми Privacy-first, навіть за умови повністю локального функціонування системи, фіксація логінів або адміністративних паролів у відкритому

					ІАЛЦ.467200.003 ПЗ	Арк.
						53
Зм.	Арк.	№ докум.	Підпис	Дата		

текстовому вигляді (plain-text) є абсолютно неприпустимою інженерною практикою.

Для ізоляції цих даних використовується алгоритм криптографічного гешування SHA-256 із обов'язковим додаванням унікальної криптографічної солі (salt). Цей математичний апарат перетворює вихідні ідентифікаційні послідовності на незворотні криптографічні геш-суми, що робить марними спроби зламу за допомогою райдужних таблиць (rainbow tables) чи словникових атак.

Реалізація такого підходу гарантовано унеможливлює компрометацію та витік бази біометричних профілів навіть за гіпотетичної умови зловмисного фізичного доступу або викрадення файлу SQLite з комп'ютера клієнта [2].

					ІАЛЦ.467200.003 ПЗ	Арк.
						54
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВОК ДО РОЗДІЛУ 2

У другому розділі пояснювальної записки було проведено комплексний системний аналіз, архітектурне проектування та науково-технічне обґрунтування вибору технологічного стека, програмних інструментів і бібліотек, які сформували основу для практичної реалізації інтелектуальної системи біометричної автентифікації на основі антропометрики обличчя. Вибір кожної технології здійснювався крізь призму критеріїв оптимізації обчислювальної складності, забезпечення криптографічної стійкості локальних даних та мінімізації часових затримок під час обробки відеопотоку у реальному часі.

Обґрунтовано доцільність застосування мови програмування Python як головного інтеграційного середовища проєкту. Незважаючи на притаманні інтерпретованим мовам обмеження швидкодії, викликані динамічною типізацією та механізмом глобального блокування інтерпретатора (GIL), вибір Python повністю виправданий його розвиненою екосистемою для комп'ютерного зору та штучного інтелекту. Ключовою перевагою є використання гібридного підходу, за якого високорівнева логіка керування застосунком та інтерфейсом взаємодіє з високоефективними низькорівневими бінарними модулями, скомпільованими мовами C та C++. Це дозволило досягти необхідного балансу між швидкістю розробки програмного продукту та його апаратною продуктивністю.

Для створення конвеєра попередньої обробки растрових графічних даних було обрано бібліотеку OpenCV. На неї покладено критично важливі інженерні задачі апаратного захоплення відеопотоку, афінних трансформацій для просторового вирівнювання положення голови користувача, а також перетворення кольірних просторів із формату BGR у стандарт RGB. Окрему увагу приділено вирішенню проблеми нестабільного зовнішнього освітлення,

					ІАЛЦ.467200.003 ПЗ	Арк.
						55
Зм.	Арк.	№ докум.	Підпис	Дата		

яка є однією з головних причин виникнення помилок першого та другого роду в біометричних системах. Інтеграція алгоритму адаптивного вирівнювання гістограми з обмеженням контрасту (CLANE) дозволила локально нормалізувати яскравість кадру, нейтралізувати жорсткі тіні та суттєво підвищити стабільність екстракції ознак.

Вирішення завдання високоточного моделювання антропометричної топології обличчя було реалізовано шляхом інтеграції фреймворку MediaPipe, а саме модуля Face Mesh. Проаналізовано його переваги над класичними монолітними згортковими неймережами (CNN). Завдяки використанню каскадної двоетапної архітектури, що складається з ультралегкого детектора BlazeFace для визначення регіону інтересу (ROI) та регресійної моделі високої щільності, система здатна обчислювати 3D-координати 468 ключових точок обличчя. Додаткова оптимізація алгоритму за рахунок оптичного трекінгу замість суцільної детекції на кожному кадрі дозволила інкапсулювати складні неймережеві обчислення у межах стандартної архітектури центрального процесора (CPU) без залучення дискретних графічних прискорювачів, забезпечуючи стабільну частоту обробки понад 30 кадрів за секунду.

Для математичного забезпечення розрахунку біометричних шаблонів було залучено бібліотеку наукових обчислень NumPy. Використання її базової структури даних ndarray дозволило організувати зберігання просторових координат у вигляді неперервних блоків оперативної пам'яті. Завдяки механізмам векторизації (Vectorization) та трансляції масивів (Broadcasting), усі лінійні трансформації, обчислення міжточкових відстаней та фінальне порівняння векторів за метриками евклідової відстані й косинусної подібності виконуються на рівні апаратних SIMD-інструкцій процесора.

З метою побудови ергономічного клієнтського застосунку та забезпечення чуйності користувацького інтерфейсу обґрунтовано вибір фреймворку PyQt. Застосування його асинхронного механізму «сигналів та слотів» у поєднанні з багатопотоковістю (QThread) дозволило успішно

					ІАЛЦ.467200.003 ПЗ	Арк.
						56
Зм.	Арк.	№ докум.	Підпис	Дата		

реалізувати патерн «Виробник-Споживач». Винесення блокуючих синхронних операцій комп'ютерного зору та нейромережевого аналізу в ізольований фоновий потік повністю ліквідувало проблему «зависання» інтерфейсу (GUI Freeze). Головний потік UI залишається виключно споживачем графічних та статусних даних, що гарантує плавність візуалізації на рівні 60 FPS та високу стабільність роботи системи при паралельних навантаженнях.

У межах проектування підсистеми збереження даних біометричних шаблонів (ембедінгів) обґрунтовано вибір вбудованої реляційної СУБД SQLite. Її безсерверна автономна архітектура дозволяє зберігати всю інформацію в єдиному локальному файлі, що ідеально відповідає концепції десктопного застосунку та повністю виключає вразливості, пов'язані з перехопленням біометричних даних під час їх транспортування мережевими каналами зв'язку. Для розширення базових можливостей СУБД та захисту інформації впроваджено модулі pickle (для бінарної серіалізації багатовимірних масивів NumPy у поля типу BLOB) та hashlib (для незворотного гешування ідентифікаторів користувачів за алгоритмом SHA-256 з додаванням криптографічної солі). Таке поєднання технологій гарантує цілісність, конфіденційність та захищеність персональних профілів навіть у випадку фізичного викрадення локального файлу бази даних.

Таким чином, сформований у другому розділі технологічний стек та розроблена багатопотокова архітектура взаємодії компонентів створили надійний, оптимізований та безпечний теоретико-технологічний фундамент. Це дозволяє безпосередньо перейти до третього розділу — детального проектування програмних компонентів, розробки структури класів та безпосередньої програмної реалізації інтелектуальної системи біометричної автентифікації.

РОЗДІЛ 3. ПРОЄКТУВАННЯ ТА ДЕТАЛІ РОЗРОБКИ ПРОГРАМНОГО КОМПЛЕКСУ

3.1 Розробка програмних компонентів обчислювального ядра

Обчислювальне ядро системи (ізольоване у пакеті *core*) є фундаментальною підсистемою застосунку, яка відповідає за математичну обробку відеопотоку в реальному часі. Архітектурно ядро декомпоновано на незалежні класи, кожен з яких виконує строго визначену задачу в загальному конвеєрі: детекцію обличчя, нормалізацію освітлення та нахилу, екстракцію математичних ознак та криптографічно захищену верифікацію.

3.1.1 Клас *FaceMeshExtractor*

Клас *FaceMeshExtractor* виступає первинним інтерфейсом взаємодії між програмним комплексом та попередньо навченою нейромережевою моделлю комп'ютерного зору. Його головне архітектурне призначення — прийом сирих відеокадрів і надточна екстракція просторової топологічної хмари, що складається з 468 антропометричних точок обличчя.

Важливим інженерним рішенням під час розробки цього компонента став перехід на сучасний стандартизований інтерфейс MediaPipe Tasks API (версії 0.10 та вище). Це дозволило отримати більш низькорівневий контроль над параметрами детектора. Програмну реалізацію конфігурації та ініціалізації графа нейромережі наведено на рисунку 3.1.

					ІАЛЦ.467200.003 ПЗ	Арк.
						58
Зм.	Арк.	№ докум.	Підпис	Дата		

```

def __init__(self, model_path=MODEL_PATH):
    download_model()

    base_options = mp_python.BaseOptions(
        model_asset_path=model_path
    )

    options = mp_vision.FaceLandmarkerOptions(
        base_options=base_options,
        running_mode=mp_vision.RunningMode.IMAGE,
        num_faces=1,
        min_face_detection_confidence=0.5,
        min_face_presence_confidence=0.5,
        min_tracking_confidence=0.5,
        output_face_blendshapes=False,
        output_facial_transformation_matrixes=False
    )

    self.landmarker = mp_vision.FaceLandmarker.create_from_options(options)
    print("FaceLandmarker initialized successfully!")

```

Рисунок 3.1 — Ініціалізація нейромережевої моделі MediaPipe FaceLandmarker

Як видно з програмного лістингу, для забезпечення повної автономності системи (безвідмовної роботи без підключення до мережі Інтернет), у модулі реалізовано автоматичне завантаження бінарного файлу ваг *face_landmarker.task*. З міркувань безпеки параметр *num_faces* жорстко обмежено до 1, щоб унеможливити авторизацію у випадку присутності сторонніх осіб у кадрі.

Головна обчислювальна логіка компонента інкапсульована у методі *process_frame*. Програмний код екстракції координат обличчя представлено на рисунку 3.2.

```

def process_frame(self, frame):
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    mp_image = mp.Image(
        image_format=mp.ImageFormat.SRGB,
        data=rgb_frame
    )

    results = self.landmarker.detect(mp_image)

    if not results.face_landmarks:
        return None

    face_landmarks = results.face_landmarks[0]

    landmarks = [
        (lm.x, lm.y, lm.z)
        for lm in face_landmarks
    ]

    return landmarks

```

Рисунок 3.2 — Програмна реалізація процесу екстракції 468 антропометричних точок

					ІАЛЦ.467200.003 ПЗ	Арк.
						59
Зм.	Арк.	№ докум.	Підпис	Дата		

Відповідно до лістингу, метод виконує апаратну конвертацію колірного простору з формату BGR у стандартний RGB та передає структуру *np.Image* до детектора. З точки зору організації пам'яті, алгоритм оптимізовано: він автоматично відкидає надлишкові метадані фреймворку (наприклад, трансформаційні матриці та *blendshapes*), конвертуючи вихідні результати у легковаговий нативний список кортежів Python (*lm.x*, *lm.y*, *lm.z*). Це суттєво зменшує навантаження на оперативну пам'ять та прискорює подальшу передачу даних конвеєром. Загальну специфікацію розроблених методів класу наведено у таблиці 3.1.

Таблиця 3.1 — Специфікація методів класу *FaceMeshExtractor*

Назва методу	Параметри	Призначення
<code>__init__</code>	<i>model_path</i>	Конфігурація порогів впевненості (Confidence) та завантаження локальної моделі
<i>process_frame</i>	<i>frame</i>	Головний метод: перетворення матриці зображення на нормалізований список 3D-координат
<i>draw_landmarks</i>	<i>frame</i> , <i>landmarks</i>	Денормалізація координат та накладання візуальної сітки точок поверх кадру користувача
<i>release</i>	Відсутні	Безпечне вивільнення ресурсів пам'яті графа обчислень при завершенні роботи

Як свідчать дані з таблиці 3.1, архітектура класу *FaceMeshExtractor* побудована за принципом суворої інкапсуляції, де всі низькорівневі операції з нейромережовим графом приховані від зовнішніх модулів. Ініціалізація важковагової моделі виконується одноразово під час створення об'єкта, що дозволяє уникнути затримок при покадровій обробці відеопотоку. Метод *process_frame* забезпечує безперервну екстракцію ключових точок, які згодом передаються на етап попередньої обробки для нормалізації.

Крім обчислювальної функції, завдяки наявності методу *draw_landmarks*, система також здатна забезпечити інтерактивний візуальний зворотний зв'язок для користувача під час реєстрації або автентифікації. Завершення роботи класу супроводжується обов'язковим викликом методу *release*, що гарантує відсутність витоків пам'яті (memory leaks) на рівні C++ ядра бібліотеки MediaPipe.

3.1.2. Клас ImagePreprocessor

Клас *ImagePreprocessor* реалізує конвеєр попередньої обробки відеокадрів перед етапом екстракції біометричного вектора ознак. Головне архітектурне завдання цього компонента — штучна стабілізація та нормалізація вхідних даних (зображення обличчя) для мінімізації впливу зовнішніх факторів, таких як нерівномірне бокове освітлення або нахил голови користувача. Впровадження цього етапу критично важливе для зниження показника помилкових відмов легітимному користувачу (FRR).

Алгоритм попередньої обробки складається з двох послідовних стадій: геометричного вирівнювання обличчя (Face Alignment) та нормалізації освітлення (CLAFHE).

Перша стадія реалізована у приватному методі *_face_alignment*. Алгоритм обчислює кут нахилу лінії очей α відносно горизонтальної осі кадру за формулою $\alpha = \arctan\left(\frac{\Delta y}{\Delta x}\right)$, використовуючи стабільні координати внутрішніх куточків очей (індекси 263 та 33). Програмну реалізацію математичного апарату афінних перетворень наведено на рисунку 3.3.

```

def _face_alignment(self, frame, landmarks):
    h, w, _ = frame.shape

    # Stable inner eye corner landmark landmarks from MediaPipe topology
    LEFT_EYE_IDX = 263
    RIGHT_EYE_IDX = 33

    lx = int(landmarks[LEFT_EYE_IDX][0] * w)
    ly = int(landmarks[LEFT_EYE_IDX][1] * h)
    rx = int(landmarks[RIGHT_EYE_IDX][0] * w)
    ry = int(landmarks[RIGHT_EYE_IDX][1] * h)

    delta_y = ry - ly
    delta_x = rx - lx

    if delta_x == 0:
        return frame

    angle_deg = math.degrees(math.atan2(delta_y, delta_x))

    # Skip transformation if tilt is negligible (< 1 deg) or extreme (> 30 deg)
    if abs(angle_deg) < 1.0 or abs(angle_deg) > 30.0:
        return frame

    center = ((lx + rx) // 2, (ly + ry) // 2)
    m_affine = cv2.getRotationMatrix2D(center, angle_deg, scale=1.0)

    # Apply warp affine transformation with mirror edge padding
    aligned_frame = cv2.warpAffine(
        frame, m_affine, (w, h),
        flags=cv2.INTER_LINEAR,
        borderMode=cv2.BORDER_REFLECT
    )

    return aligned_frame

```

Рисунок 3.3 — Реалізація геометричного вирівнювання обличчя через афінне перетворення простору

Як показано на рисунку 3.3, після обчислення кута формується матриця двовимірної повороту за допомогою функції `cv2.getRotationMatrix2D`, після чого весь кадр розгортається у фронтальну площину методом `cv2.warpAffine`.

Друга стадія обробки полягає у застосуванні адаптивного вирівнювання гістограми з обмеженням контрасту (CLAHE). Логіку роботи цього етапу наведено на рисунку 3.4.

```

def _apply_clahe(self, frame):
    lab = cv2.cvtColor(frame, cv2.COLOR_BGR2LAB)
    l_channel, a_channel, b_channel = cv2.split(lab)

    l_clahe = self.clahe.apply(l_channel)
    lab_clahe = cv2.merge([l_clahe, a_channel, b_channel])

    return cv2.cvtColor(lab_clahe, cv2.COLOR_LAB2BGR)

```

Рисунок 3.4 — Адаптивна нормалізація освітлення у колірному просторі LAB

Відповідно до лістингу, застосування алгоритму CLAHE безпосередньо до каналів BGR призвело б до серйозних колірних спотворень. Тому інженерним рішенням стала конвертація кадру у простір LAB, де алгоритм застосовується виключно до каналу L (яскравість), залишаючи канали A та B (колірні складові) недоторканими. Це дозволяє висвітлити затінені ділянки обличчя, зберігаючи при цьому природний колір шкіри. Загальну специфікацію всіх розроблених методів класу наведено у таблиці 3.2.

Таблиця 3.2 — Специфікація методів класу *ImagePreprocessor*

Назва методу	Параметри	Призначення
<code>__init__</code>	<code>clahe_clip_limit</code> , <code>clahe_tile_size</code>	Ініціалізація об'єкта обробки CLAHE із заданим порогом зрізу контрасту та розміром локального тайлу (за замовчуванням 8×8 пікселів)
<code>process</code>	<code>frame</code> , <code>landmarks</code>	Публічний метод-оркестратор. Викликає послідовний конвеєр: вирівнювання обличчя (якщо точки наявні), нормалізація освітлення, повернення готового кадру
<code>_face_alignment</code>	<code>frame</code> , <code>landmarks</code>	Приватний метод для компенсації нахилу голови користувача за допомогою матриць афінного перетворення OpenCV
<code>_apply_clahe</code>	<code>frame</code>	Приватний метод для вирівнювання тіней та оптимізації освітленості за допомогою маніпуляцій з яскравістю у колірному просторі LAB

Аналізуючи програмну структуру класу *ImagePreprocessor*, варто відзначити суворе дотримання принципів об'єктно-орієнтованого програмування: складна математична логіка прихована у приватних методах, тоді як зовнішнім модулям доступний лише єдиний інтерфейс *process*.

Окремої уваги заслуговує програмний запобіжник у методі вирівнювання обличчя: система ігнорує мікронахили (менше 1°) для заощадження ресурсів процесора, а також блокує екстремальні нахили (понад 30°), оскільки такі афінні перетворення призводять до критичної втрати піксельної інформації на краях зображення.

Це свідомий архітектурний компроміс між стійкістю до поворотів та швидкодією системи.

3.1.3. Клас *AnthropometryAnalyzer*

Клас *AnthropometryAnalyzer* є математичним ядром системи, яке відповідає за перетворення сирого масиву просторових координат (468 точок) у компактний, нормалізований та математично інваріантний вектор біометричних ознак (ембеддінг).

Головна архітектурна мета цього компонента — звести складну 3D-топологію обличчя до 20 ключових пропорцій, які залишаються незмінними незалежно від міміки користувача.

Важливим етапом ініціалізації аналізатора є визначення антропометричних констант та стабільних пар точок. Програмну реалізацію конструктора класу наведено на рисунку 3.5.

					ІАЛЦ.467200.003 ПЗ	Арк.
						64
Зм.	Арк.	№ докум.	Підпис	Дата		

```

def __init__(self):

    # Baseline single points for normalization (inner corners of the eyes)
    self.LEFT_INNER_EYE = 362
    self.RIGHT_INNER_EYE = 133

    # Landmark pairs selected for stable geometric facial proportions (index_A, index_B, name)
    self.LANDMARK_PAIRS = [
        # --- Eye Region ---
        (362, 263, 'left_eye_width'),
        (33, 133, 'right_eye_width'),
        (159, 145, 'left_eye_height'),
        (386, 374, 'right_eye_height'),
        (362, 133, 'interpupillary_distance'),

        # --- Nose Region ---
        (1, 168, 'nose_bridge_height'),
        (129, 358, 'nose_width'),
        (168, 6, 'nose_ridge_length'),
        (1, 2, 'nose_tip_vertical'),

        # --- Inter-regional Proportions ---
        (168, 0, 'bridge_to_upper_lip'),
        (362, 1, 'left_eye_to_nose'),
        (133, 1, 'right_eye_to_nose'),
        (1, 17, 'nose_to_chin'),
        (10, 152, 'face_height'),
        (234, 454, 'face_width'),

        # --- Lip Region ---
        (61, 291, 'mouth_width'),
        (0, 17, 'mouth_height'),
        (13, 14, 'inter_lip_distance'),

        # --- Eyebrow Region ---
        (70, 300, 'inter_eyebrow_width'),
        (105, 334, 'eyebrow_to_eye_height'),
    ]

```

Рисунок 3.5 — Ініціалізація антропометричних констант та стабільних геометричних пар обличчя

Як показано на програмному лістингу, для розрахунку базового масштабовального інтервалу було обрано внутрішні куточки очей (індекси 362 та 133). З анатомічної точки зору це найбільш стабільні точки обличчя, просторове положення яких практично не піддається деформації під впливом м'язів (наприклад, під час посмішки чи моргання). Масив *LANDMARK_PAIRS* містить 20 ретельно відібраних зв'язків, які описують унікальні пропорції між різними регіонами обличчя (зони очей, носа, губ, брів).

Безпосередня генерація біометричного вектора реалізована через методи обчислення евклідової відстані та нормалізації. Код цих обчислювальних методів представлено на рисунку 3.6.

					ІАЛЦ.467200.003 ПЗ	Арк.
						65
Зм.	Арк.	№ докум.	Підпис	Дата		

```

def _euclidean_distance(self, p1, p2):
    return math.sqrt(
        (p1[0] - p2[0]) ** 2 +
        (p1[1] - p2[1]) ** 2 +
        (p1[2] - p2[2]) ** 2
    )

def extract_embedding(self, landmarks):
    if landmarks is None or len(landmarks) < 468:
        return None

    # Step 1: Calculate baseline normalizing distance (dbase)
    p_left = landmarks[self.LEFT_INNER_EYE]
    p_right = landmarks[self.RIGHT_INNER_EYE]
    d_base = self._euclidean_distance(p_left, p_right)

    # Division by zero protection
    if d_base < 1e-6:
        return None

    # Steps 2 & 3: Calculate distances and normalize (fk = d(Pi, Pj) / dbase)
    feature_vector = []
    for (idx_a, idx_b, _) in self.LANDMARK_PAIRS:
        pa = landmarks[idx_a]
        pb = landmarks[idx_b]
        distance = self._euclidean_distance(pa, pb)
        feature_vector.append(distance / d_base)

    # Step 4: Return as numpy array for fast vector operations
    return np.array(feature_vector, dtype=np.float32)

```

Рисунок 3.6 — Алгоритм екстракції та нормалізації багатовимірного вектора ознак

Відповідно до рисунка 3.6, для кожної пари точок P_i та P_j обчислюється абсолютна лінійна відстань у тривимірному просторі за формулою Евклідової метрики:

$$d(P_i, P_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

Для того щоб система коректно розпізнавала користувача незалежно від його відстані до об'єктива камери (масштабна інваріантність), абсолютні відстані підлягають обов'язковій нормалізації. Нормалізоване значення кожної ознаки f_k розраховується як відношення локальної відстані до базового міжзіничного інтервалу d_{base} :

$$f_k = \frac{d(P_i, P_j)}{d_{base}}$$

У фіналі метод формує оптимізований одновимірний масив типу `numpy.ndarray`, готовий до швидких матричних операцій. Детальну специфікацію методів класу наведено у таблиці 3.3.

Таблиця 3.3 — Специфікація методів класу *AnthropometryAnalyzer*

Назва методу	Параметри	Призначення
<code>__init__</code>	Відсутні	Оголошення топологічних індексів MediaPipe та формування списку з 20 стабільних пар точок
<code>_euclidean_distance</code>	$p1, p2$	Приватний метод розрахунку абсолютної лінійної відстані між двома 3D-координатами
<code>extract_embedding</code>	<i>landmarks</i>	Головний метод: обчислення базової нормалізуючої відстані, ітеративний розрахунок дистанцій для пар, нормалізація та повернення масиву NumPy форми [20]
<code>get_feature_names</code>	Відсутні	Допоміжний метод для отримання текстових дескрипторів ознак (використовується для діагностики та ведення логів)

Як свідчить аналіз наведеної архітектури, впровадження класу *AnthropometryAnalyzer* вирішує критичну проблему надмірної розмірності даних (Dimensionality Reduction). Замість обробки 1404 координат (468 точок у 3D-вимірах), які постійно флюктуюють залежно від ракурсу та освітлення, система оперує легковаговим інваріантним вектором із 20 дійсних чисел (тип `float32`).

Застосування бібліотеки NumPy гарантує, що подальша серіалізація та матричні порівняння векторів у базі даних виконуватимуться за частки мілісекунди, що повністю задовольняє вимоги до систем реального часу. Наявність захисту від ділення на нуль (перевірка $d_{base} < 1e - 6$)

унеможливилює аварійне завершення програми у випадках, коли обличчя користувача знаходиться занадто близько або занадто далеко від фокусної площини камери.

3.1.4. Клас *BiometricMatcher*

Клас *BiometricMatcher* виступає алгоритмічним модулем верифікації, що відповідає за прийняття остаточного рішення щодо допуску користувача. Його головна функція — математичне порівняння поточного вектора ознак, отриманого з відеокамери, зі збереженим еталонним профілем із бази даних.

Інноваційним інженерним рішенням у даній роботі є відмова від використання єдиної метрики порівняння на користь двофакторної гібридної моделі. Для обчислення ступеня збігу векторів застосовуються одночасно Евклідова відстань та Косинусна подібність. Програмну реалізацію математичного апарату цих метрик наведено на рисунку 3.7.

```
def euclidean_distance(self, vec1, vec2):  
    return float(np.linalg.norm(vec1 - vec2))  
  
def cosine_similarity(self, vec1, vec2):  
    norm1 = np.linalg.norm(vec1)  
    norm2 = np.linalg.norm(vec2)  
  
    if norm1 < 1e-6 or norm2 < 1e-6:  
        return 0.0  
  
    return float(np.dot(vec1, vec2) / (norm1 * norm2))
```

Рисунок 3.7 — Програмна реалізація математичних метрик порівняння багатовимірних векторів

Згідно з лістингом коду, перша метрика — Евклідова відстань — визначає абсолютну лінійну дистанцію між кінцями двох векторів U та V у M -вимірному просторі за формулою:

$$D_E(U, V) = \sqrt{\sum_{k=1}^M (f_k - f'_k)^2}$$

Цей параметр ефективно контролює лінійну відповідність антропометричних пропорцій і є критично важливим для мінімізації коефіцієнта помилкового доступу (FAR).

Друга метрика — Косинусна подібність — оцінює косинус кута між двома векторами, ігноруючи їхні абсолютні модулі:

$$S_C(U, V) = \frac{U \cdot V}{\|U\| \|V\|}$$

Ця метрика компенсує лінійні флуктуації координат, які виникають при зміні ракурсу або мікронахилах голови. Значення наближається до 1.0 за умови абсолютної колінеарності (збігу пропорцій).

Логіка прийняття рішення інкапсульована у методі *verify*, реалізацію якого представлено на рисунку 3.8.

```
def verify(self, current_vector, saved_vector):
    if current_vector is None or saved_vector is None:
        return {
            'authenticated': False,
            'euclidean' : None,
            'cosine' : None,
            'eu_passed' : False,
            'cos_passed' : False,
        }

    eu_dist = self.euclidean_distance(current_vector, saved_vector)
    cos_sim = self.cosine_similarity(current_vector, saved_vector)

    eu_passed = eu_dist <= self.euclidean_threshold
    cos_passed = cos_sim >= self.cosine_threshold

    authenticated = eu_passed and cos_passed

    return {
        'authenticated': authenticated,
        'euclidean' : round(eu_dist, 4),
        'cosine' : round(cos_sim, 4),
        'eu_passed' : eu_passed,
        'cos_passed' : cos_passed,
    }
```

Рисунок 3.8 — Двофакторна логіка верифікації біометричних шаблонів

Відповідно до рисунка 3.8, метод *verify* реалізує сувору кон'юнктивну логіку (оператор *AND*): доступ надається виключно в тому випадку, коли обидві

метрики одночасно задовольняють встановлені безпекові пороги (за замовчуванням ≤ 0.35 для Евклідової відстані та ≥ 0.92 для Косинусної подібності). Загальну специфікацію всіх розроблених методів класу наведено у таблиці 3.4.

Таблиця 3.4 — Специфікація методів класу *BiometricMatcher*

Назва методу	Параметри	Призначення
<code>__init__</code>	<code>euclidean_threshold</code> , <code>cosine_threshold</code>	Ініціалізація екземпляра класу із заданими гіперпараметрами допуску
<code>euclidean_distance</code>	<code>vec1</code> , <code>vec2</code>	Обчислення лінійної евклідової відстані за допомогою функції <code>numpy.linalg.norm</code>
<code>cosine_similarity</code>	<code>vec1</code> , <code>vec2</code>	Обчислення скалярного добутку векторів, поділеного на добуток їхніх норм
<code>verify</code>	<code>current_vector</code> , <code>saved_vector</code>	Комплексна перевірка. Повертає словник із булевим статусом допуску (<i>authenticated</i>) та обчисленими значеннями метрик
<code>update_thresholds</code>	<code>euclidean_threshold</code> , <code>cosine_threshold</code>	Динамічне оновлення порогів під час виконання програми (runtime) без необхідності перезавантаження ядра

Аналіз реалізованого класу підтверджує високу гнучкість системи. Замість повернення простого булевого значення *True/False*, метод *verify* формує структурований словник *dict* із детальними аналітичними метриками. Це архітектурне рішення забезпечує безперешкодну інтеграцію модуля з користувацьким інтерфейсом для відображення статистики в реальному часі, а також дозволяє передавати точні математичні показники до модуля *DatabaseManager* для їх подальшого збереження у журналі аудиту (*AuthLog*). Крім того, наявність методу *update_thresholds* дозволяє динамічно калібрувати чутливість системи з графічного інтерфейсу, адаптуючи її до різних умов освітленості без зупинки процесу захоплення кадрів.

3.1.5. Клас *SecurityManager*

Клас *SecurityManager* є спеціалізованим криптографічним модулем обчислювального ядра, який відповідає за безпеку облікових даних користувачів.

Відповідно до сучасної архітектурної парадигми Privacy-first, система спроектована таким чином, що жодні персональні дані (ні логіни, ні паролі) не зберігаються у базі даних у відкритому вигляді (plain-text).

Головним завданням класу є забезпечення стійкості системи до класичних векторів кібератак, зокрема атак за словником (Dictionary attacks) та використання райдужних таблиць (Rainbow tables).

Для реалізації цього механізму застосовується криптографічне гешування за стандартом SHA-256 із додаванням унікальної випадкової солі (salt). Програмну реалізацію генерації захищеного шаблону наведено на рисунку 3.9.

```
def hash_password(self, password):  
  
    salt = os.urandom(self.SALT_SIZE)  
    salted = salt + password.encode(self.ENCODING)  
    hash_bytes = hashlib.sha256(salted).digest()  
  
    return f"{salt.hex()}{self.SEPARATOR}{hash_bytes.hex()}"
```

Рисунок 3.9 — Програмна реалізація алгоритму гешування з використанням криптографічної солі

Як свідчить лістинг коду, під час реєстрації система генерує 16-байтну криптографічно стійку псевдовипадкову послідовність за допомогою системного виклику *os.urandom*. Ця сіль конкатенується з паролем користувача, після чого від утвореного рядка обчислюється хеш-сума SHA-256.

У базу даних записується складений рядок формату *hex_salt\$hex_hash*, де сіль та геш розділені спеціальним символом. Такий підхід гарантує, що навіть

					ІАЛЦ.467200.003 ПЗ	Арк.
						71
Зм.	Арк.	№ докум.	Підпис	Дата		

для однакових паролів різних користувачів у базі даних будуть збережені абсолютно різні хеш-суми.

Не менш важливою є логіка зворотної перевірки (верифікації) під час спроби доступу, яку зображено на рисунку 3.10.

```
def verify_password(self, password, stored_hash):  
  
    try:  
        parts = stored_hash.split(self.SEPARATOR)  
        if len(parts) != 2:  
            return False  
  
        salt_hex, hash_hex = parts  
        salt = bytes.fromhex(salt_hex)  
  
        salted = salt + password.encode(self.ENCODING)  
        hash_bytes = hashlib.sha256(salted).digest()  
  
        # Uses time-constant comparison to eliminate timing attack vectors  
        return hmac.compare_digest(hash_bytes.hex(), hash_hex)  
  
    except (ValueError, AttributeError):  
        return False
```

Рисунок 3.10 — Алгоритм безпечної верифікації облікових даних за константний час

Відповідно до рисунка 3.10, метод вилучає сіль зі збереженого рядка, повторно гешує вхідний пароль і порівнює результати. Важливим інженерним рішенням на цьому етапі є відмова від стандартного оператора порівняння рядків (==) на користь функції *hmac.compare_digest*.

Це нівелює ризик виникнення атак за часом (Timing attacks), оскільки операція порівняння завжди виконується за константний час, незалежно від того, на якому символі знайдено розбіжність.

Загальну специфікацію всіх розроблених методів класу наведено у таблиці 3.5.

					ІАЛЦ.467200.003 ПЗ	Арк.
						72
Зм.	Арк.	№ докум.	Підпис	Дата		

Таблиця 3.5 — Специфікація методів класу *SecurityManager*

Назва методу	Параметри	Призначення
<i>hash_password</i>	<i>password</i>	Створення захищеного запису шляхом генерації унікальної 16-байтної солі та обчислення SHA-256 гешу
<i>verify_password</i>	<i>password</i> , <i>stored_hash</i>	Безпечна перевірка вхідного пароля відносно еталонного гешу за допомогою <i>hmac.compare_digest</i>
<i>hash_username</i>	<i>username</i>	Детерміноване гешування ідентифікатора користувача без використання солі для створення унікального ключа пошуку в базі даних

Аналізуючи структуру компонента *SecurityManager*, окремо слід виділити метод *hash_username*. На відміну від паролів, імена користувачів (або їхні ідентифікатори) потребують детермінованого гешування (однаковий вхід завжди дає однаковий хеш). Це дозволяє системі швидко знаходити користувача в базі даних за допомогою SQL-запитів *SELECT ... WHERE username = ?*, при цьому саме ім'я користувача залишається прихованим від адміністратора бази даних або злоумисника у випадку витоку файлу бази даних (Data Breach). Таким чином, клас забезпечує фундаментальний рівень криптографічного захисту всієї програмної інфраструктури.

3.1.6. Клас *TextureLivenessDetector*

Для забезпечення первісного інфраструктурного захисту біометричного конвеєра від статичних атак підміни (2D Spoofing Attacks) за допомогою роздрукованих високоякісних фотокарток або екранів портативних цифрових пристроїв (моніторів, смартфонів) було спроектовано та програмно реалізовано клас *TextureLivenessDetector*.

Даний компонент функціонує в пасивному режимі й аналізує структурні та спектральні властивості відеокадру на піксельному рівні, не вимагаючи від користувача виконання інтерактивних дій.

Аналіз дисперсії оператора Лапласа (Laplacian Variance) використовується для детекції дефіциту мікротекстурної чіткості. Оскільки повторно захоплені зображення з екранів або паперу проходять через подвійний оптичний конвеєр (камера запису, далі екран відтворення, далі вебкамера системи), вони втрачають високі частоти спектра. Оператор Лапласа обчислює другу просторову похідну яскравості зображення:

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Дисперсія цього оператора дозволяє формалізувати ступінь розмиття або наявності артефактів цифрового перевідбиття. Емпірично встановлений та адаптований під умови кімнатного освітлення поріг чіткості визначено на рівні \$30.0\$. Значення нижче цього порогу інтерпретуються системою як аномалія текстури (Blur Anomaly).

Цифрові екрани випромінюють активне світло (емісійний профіль), що створює неприродні сплески насиченості в каналах Cr (компонента червоної різниці) та Cb (компонента синьої різниці), які кардинально відрізняються від характеристик відбиття світла живою людською шкірою. Клас виділяє зону обличчя (Face Crop), конвертує її з простору BGR у YCrCb та розраховує середнє значення каналів хроматину. Для природної шкіри в умовах вебкамери встановлено строгі обмеження:

$$125 \leq \mu_{Cr} \leq 185 \quad \text{та} \quad 70 \leq \mu_{Cb} \leq 135$$

Вихід за межі цих інтервалів сигналізує про колірну аномалію (Color Distribution Anomaly). Клас повертає результуючий прапор *is_spoof*. Переглянути результати роботи можна на рисунку 3.11.

					ІАЛЦ.467200.003 ПЗ	Арк.
						74
Зм.	Арк.	№ докум.	Підпис	Дата		

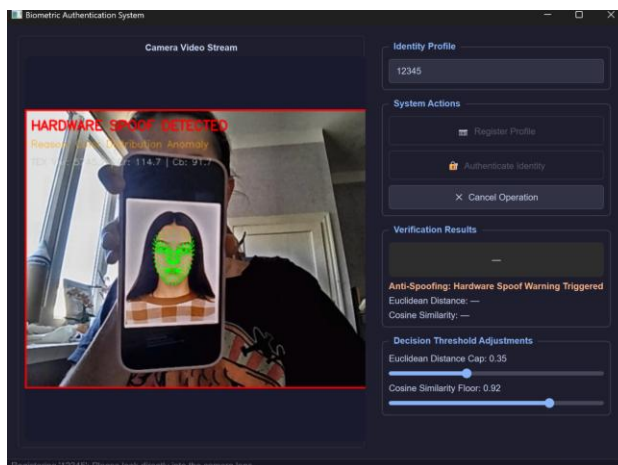


Рисунок 3.11 — Візуалізація процесу двофакторної перевірки живості особи (Anti-Spoofing) обчислювальним ядром системи

Якщо зафіксовано бодай одну з аномалій, обчислювальний конвеєр миттєво блокується з виведенням повідомлення "HARDWARE SPOOF DETECTED".

3.1.7. Клас *LivenessDetector*

Для нейтралізації складних динамічних атак типу відтворення відеозапису (Replay Attacks), коли зловмисник демонструє вебкамері заздалегідь записане відео з морганням та рухами легітимного користувача, було спроектовано клас *LivenessDetector*.

Замість випадкової перевірки однієї ознаки, система вимагає від об'єкта перед камерою суворого послідовного виконання чотирьох біометричних мікрооперацій:

1. Первинне фіксування закриття повік (Blink 1).
2. Мікроповорот голови ліворуч (Turn Left).
3. Мікроповорот голови праворуч (Turn Right).
4. Вторинне фіксування закриття повік (Blink 2).

Верифікація моргання здійснюється через метрику EAR (Eye Aspect Ratio), що визначає відношення вертикальних відстаней між повіками до горизонтальної ширини ока. Поріг моргання встановлено на рівні 0.23. Коли поточний показник падає нижче порогу, фіксується закриття ока, а при поверненні в норму — повний цикл моргання.

Верифікація тривимірного повороту голови (Yaw Ratio) базується на антропометричних пропорціях. Клас обчислює евклідову відстань від кінчика носа (*NOSE_TIP*, індекс 1) до крайньої лівої точки обличчя (*LEFT_CHEEK*, індекс 234) та крайньої правої точки (*RIGHT_CHEEK*, індекс 454). Розраховується коефіцієнт асиметрії:

$$Yaw\ Ratio = \frac{Distance(Nose, Left\ Cheek)}{Distance(Nose, Right\ Cheek)}$$

При повороті голови вліво значення падає нижче порогу 0.92, при повороті вправо — перевищує 1.10. Оскільки плоска фотокартка або екран телефону при обертанні в просторі дають лінійне стиснення всієї проекції, деформація пропорцій Yaw Ratio не відбувається, що дозволяє математично точно відсікати 2D-носії. Переглянути результати роботи можна на рисунку 3.12.

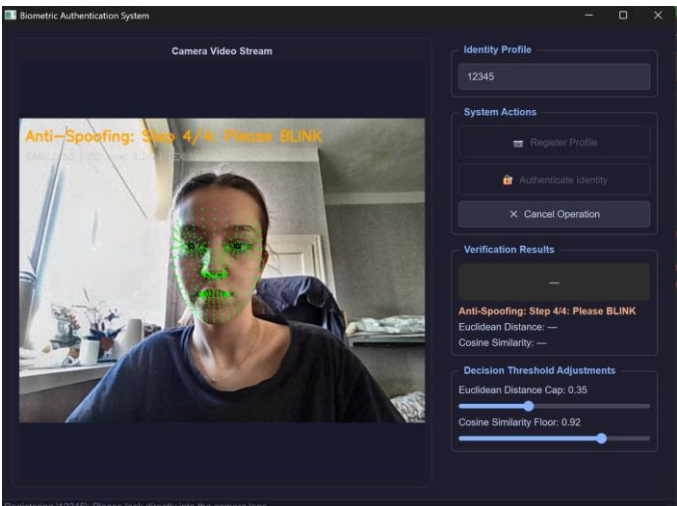


Рисунок 3.12 — Графічний інтерфейс користувача (GUI) інтелектуальної системи у робочому стані

Компонент динамічно передає інструкції поточного кроку конвеєра в інтерфейс програми через атрибут *instruction* та утримує прапор *is_live* у стані *False* до моменту успішного завершення останнього етапу автомата станів.

3.2. Проєктні рішення підсистеми збереження даних

Підсистема збереження даних є критичним інфраструктурним компонентом розробленого програмного комплексу. З огляду на вимоги щодо автономності роботи системи та недопущення витоку біометричної інформації через мережу Інтернет, було прийнято рішення відмовитися від хмарних рішень на користь локальної безсерверної реляційної системи управління базами даних (СУБД) SQLite. Увесь масив даних зберігається в єдиному зашифрованому локальному файлі, доступ до якого суворо інкапсульований у межах розробленого пакета *database*.

3.2.1. Клас *DatabaseManager*

Клас *DatabaseManager* реалізує патерн доступу до даних (Data Access Object, DAO) та виступає єдиним програмним інтерфейсом між обчислювальним ядром застосунку, графічним інтерфейсом та фізичним файлом бази даних SQLite. Головне архітектурне призначення цього класу — абстрагування SQL-запитів, управління пулом з'єднань та забезпечення потокобезпечних (Thread-safe) транзакцій.

Важливим інженерним рішенням на етапі ініціалізації бази даних стала адаптація з'єднання до багатопотокової архітектури системи. Програмну реалізацію конструктора класу наведено на рисунку 3.13.

					ІАЛЦ.467200.003 ПЗ	Арк.
						77
Зм.	Арк.	№ докум.	Підпис	Дата		

```
def __init__(self, db_path=DEFAULT_DB_PATH):
    self.db_path = db_path
    self.security = SecurityManager()

    self._connection = sqlite3.connect(self.db_path, check_same_thread=False)
    self._create_tables()
    print(f"Database connected successfully at: {self.db_path}")
```

Рисунок 3.13 — Ініціалізація підключення до локальної СУБД SQLite

Як видно з програмного лістингу, підключення до бази даних ініціалізується з параметром *check_same_thread=False*. Оскільки в розробленій системі графічний потік (UI Thread) та фоновий обчислювальний потік (Worker Thread) звертаються до бази даних асинхронно, вимкнення цієї перевірки є критично необхідним для уникнення помилок конкурентного доступу з боку ядра SQLite. Також у конструкторі відбувається композиція з класом *SecurityManager*, що гарантує криптографічну обробку всіх вхідних ідентифікаторів.

Головна логіка запису та читання біометричних профілів реалізована у методах *save_user* та *get_user*. Фрагмент коду, що відповідає за реєстрацію нового користувача, представлено на рисунку 3.14.

```
def save_user(self, display_name, embedding):

    username_hash = self.security.hash_username(display_name.lower().strip())
    embedding_blob = pickle.dumps(embedding)

    try:
        cursor = self._connection.cursor()
        cursor.execute("""
            INSERT INTO Users (username, display_name, embedding)
            VALUES (?, ?, ?)
        """, (username_hash, display_name, embedding_blob))
        self._connection.commit()
        print(f"User profile '{display_name}' saved successfully.")
        return True
    except sqlite3.IntegrityError:
        print(f"User verification error: '{display_name}' already registered.")
        return False
```

Рисунок 3.14 — Програмна логіка безпечного збереження біометричного еталона

					ІАЛЦ.467200.003 ПЗ	Арк.
						78
Зм.	Арк.	№ докум.	Підпис	Дата		

Відповідно до рисунка 3.14, перед виконанням транзакції ім'я користувача проходить детерміноване гешування (відповідно до парадигми Privacy-first). З метою захисту від атак типу SQL-ін'єкцій (SQL Injection), усі запити до бази даних виконуються виключно за допомогою параметризованих інструкцій (через плейсхолдери ?).

Крім того, на рівні бази даних обробляється виняток *sqlite3.IntegrityError*, який виникає у разі спроби повторної реєстрації користувача з уже існуючим ідентифікатором (завдяки обмеженню *UNIQUE* у схемі таблиці). Загальну специфікацію розроблених методів класу наведено у таблиці 3.6.

Таблиця 3.6 — Специфікація основних методів класу *DatabaseManager*

Назва методу	Параметри	Призначення
<code>__init__</code>	<code>db_path</code>	Встановлення з'єднання з базою даних, виклик автоматичної міграції структури таблиць
<code>_create_tables</code>	Відсутні	Приватний метод для генерації реляційної схеми таблиць <i>Users</i> та <i>AuthLog</i> (якщо вони не існують)
<code>save_user</code>	<code>display_name</code> , <code>embedding</code>	Гешування логіну, бінарна серіалізація вектора ознак та параметризований запис <i>INSERT</i> у БД
<code>get_user</code>	<code>display_name</code>	Виконання запиту <i>SELECT</i> , десеріалізація бінарних даних та повернення об'єкта <i>numpy.ndarray</i>
<code>log_auth_attempt</code>	<code>display_name</code> , <code>success</code> , <code>euclidean</code> , <code>cosine</code>	Запис телеметрії та результату верифікації у журнал математичного аудиту <i>AuthLog</i>
<code>close</code>	Відсутні	Безпечне вивільнення файлового дескриптора бази даних (запобігання блокуванню файлу <i>face_auth.db</i>)

Аналізуючи структуру класу *DatabaseManager*, можна стверджувати, що він повністю інкапсулює SQL-логіку. Зовнішні модулі програми (такі як *VideoThread* або *MainWindow*) взаємодіють із базою даних виключно через високорівневі методи Python, передаючи об'єкти та отримуючи готові структури даних (словники або масиви NumPy). Це суттєво знижує зв'язність коду (Coupling) та підвищує його тестованість. Наявність вбудованого журналу аудиту (*log_auth_attempt*) забезпечує ведення безперервної історії подій безпеки без потреби залучення сторонніх систем логування.

3.2.2. Фізичне проєктування структури таблиць *Users* та *AuthLog*

Фізичне проєктування бази даних визначає, яким чином логічні моделі даних відображаються на структури збереження обраної СУБД (SQLite). У розробленій системі архітектура даних складається з двох взаємопов'язаних відношень (таблиць): *Users* (для зберігання біометричних еталонів та ідентифікаторів) та *AuthLog* (для ведення безперервного журналу математичного аудиту доступу).

Створення реляційної схеми ініціюється автоматично під час першого запуску системи. Програмну реалізацію SQL-запитів DDL (Data Definition Language) для створення таблиць наведено на рисунку 3.15.

```

def _create_tables(self):

    cursor = self._connection.cursor()

    # Users profile storage
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS Users (
            id            INTEGER PRIMARY KEY AUTOINCREMENT,
            username      TEXT UNIQUE NOT NULL,
            display_name   TEXT NOT NULL,
            embedding      BLOB NOT NULL,
            created_at     TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        )
    """)

    # System security and mathematical audit logs
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS AuthLog (
            id            INTEGER PRIMARY KEY AUTOINCREMENT,
            display_name   TEXT NOT NULL,
            success        INTEGER NOT NULL,
            euclidean      REAL,
            cosine         REAL,
            attempted_at   TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        )
    """)

    self._connection.commit()

```

Рисунок 3.15 — SQL-інструкції визначення фізичної схеми бази даних системи

Головною інженерною проблемою під час проєктування таблиці *Users* стала відсутність у СУБД SQLite нативного типу даних для збереження багатовимірних масивів (на кшталт масивів *numpy.ndarray*, які генерує математичне ядро). Для вирішення цієї проблеми технологічний стек було доповнено стандартним модулем Python *pickle*, який забезпечує глибоку бінарну серіалізацію об'єктів.

Перед записом еталонного профілю в базу даних вектор ознак обробляється функцією *pickle.dumps()*, перетворюючись на безперервний потік байтів (byte-stream). Цей потік зберігається у колонці типу *BLOB* (Binary Large Object). Під час процесу автентифікації зворотний метод *pickle.loads()* за доли мілісекунд відновлює повноцінний об'єкт у оперативній пам'яті, суворо зберігаючи початкові типи даних (наприклад, масив *float32*) та забезпечуючи їхню готовність до векторизованих матричних розрахунків. Детальну специфікацію полів таблиці *Users* наведено у таблиці 3.7.

					ІАЛЦ.467200.003 ПЗ	Арк.
						81
Зм.	Арк.	№ докум.	Підпис	Дата		

Таблиця 3.7 — Фізична структура таблиці *Users*

Назва поля	Тип даних SQLite	Обмеження	Призначення
<i>id</i>	<i>INTEGER</i>	<i>PRIMARY KEY AUTOINCREMENT</i>	Суррогатний первинний ключ для внутрішньої ідентифікації запису
<i>username</i>	<i>TEXT</i>	<i>UNIQUE NOT NULL</i>	Детермінований SHA-256 геш імені користувача. Використовується як індекс пошуку
<i>display_name</i>	<i>TEXT</i>	<i>NOT NULL</i>	Відображуване ім'я користувача у відкритому вигляді (для інтерфейсу)
<i>embedding</i>	<i>BLOB</i>	<i>NOT NULL</i>	Серіалізований через <i>pickle</i> біометричний масив <i>numpy.ndarray</i>
<i>created_at</i>	<i>TIMESTAMP</i>	<i>DEFAULT CURRENT_TIMEST AMP</i>	Час та дата реєстрації біометричного профілю в системі

Другою складовою реляційної схеми є таблиця *AuthLog*, яка реалізує функціонал математичного аудиту. Відповідно до корпоративних стандартів розробки систем контролю та управління доступом (СКУД), система повинна фіксувати не лише факт входу, але й телеметрію прийняття рішень. Тому таблиця зберігає обчислені значення Евклідової відстані та Косинусної подібності для кожної спроби. Це дозволяє в майбутньому аналізувати стабільність системи, розраховувати похибки FAR/FRR та виявляти аномалії (наприклад, спроби підробки обличчя). Детальну специфікацію полів таблиці *AuthLog* наведено у таблиці 3.8.

Таблиця 3.8 — Фізична структура таблиці *AuthLog*

Назва поля	Тип даних SQLite	Обмеження	Призначення
<i>id</i>	<i>INTEGER</i>	<i>PRIMARY KEY</i> <i>AUTOINCREMENT</i>	Унікальний ідентифікатор транзакції доступу
<i>display_name</i>	<i>TEXT</i>	<i>NOT NULL</i>	Ім'я користувача, для якого ініційовано верифікацію
<i>success</i>	<i>INTEGER</i>	<i>NOT NULL</i>	Булевий прапорець (1 — доступ дозволено, 0 — відхилено)
<i>euclidean</i>	<i>REAL</i>	З допусканням <i>NULL</i>	Обчислене значення лінійної метрики (Евклідової відстані)
<i>cosine</i>	<i>REAL</i>	З допусканням <i>NULL</i>	Обчислене значення кутової метрики (Косинусної подібності)
<i>attempted_at</i>	<i>TIMESTAMP</i>	<i>DEFAULT</i> <i>CURRENT_TIMESTAMP</i>	Точний час фіксації спроби автентифікації

Спроектвана реляційна схема знаходиться у третій нормальній формі (3NF), що унеможливорює виникнення аномалій оновлення чи дублювання даних. Використання модифікатора *UNIQUE* для поля *username* переносить відповідальність за унікальність профілів з рівня застосунку на рівень рушія СУБД, що гарантує цілісність даних навіть у випадку конкурентних запитів.

3.3. Реалізація компонентів користувацького інтерфейсу

Користувацький інтерфейс розробленої системи біометричної автентифікації побудований на базі сучасного фреймворку *PySide6* (офіційного порту Qt 6 для мови Python). Головним архітектурним викликом під час проєктування графічної оболонки для систем комп'ютерного зору реального часу є запобігання блокуванню головного потоку виконання програми (Main

Thread). Оскільки алгоритми нейромережевої екстракції точок (MediaPipe) та матричні розрахунки (NumPy) є ресурсомісткими процесами, їх виконання безпосередньо у потоці інтерфейсу неминуче призвело б до явища «зависання» графічних віджетів (GUI Freeze).

Для розв'язання цієї проблеми архітектура пакета *gui* була декомпонована за класичним патерном «Виробник-Споживач» (Producer-Consumer).

3.3.1. Патерн асинхронного виконання на базі класу *VideoThread*

Клас *VideoThread* є програмним втіленням сутності «Виробник» (Producer) у розробленому архітектурному патерні. Він успадковується від базового класу *QThread* і є повністю ізольованим контекстом виконання на рівні операційної системи.

Його головне завдання — захоплення відеокадрів, управління обчислювальним ядром та трансляція готових результатів до головного вікна програми.

Взаємодія між фоновим робочим потоком (*VideoThread*) та головним потоком інтерфейсу здійснюється виключно за допомогою асинхронного механізму сигналів та слотів (Signals and Slots), який гарантує сувору потокобезпеку (Thread Safety).

Програмну реалізацію сигналів міжпотокової взаємодії наведено на рисунку 3.16.

```
frame_ready    = Signal(QImage)
auth_result    = Signal(bool, float, float)
status_message = Signal(str)
error_occurred = Signal(str)

MODE_IDLE      = "idle"
MODE_REGISTER  = "register"
MODE_AUTH      = "auth"
```

Рисунок 3.16 — Асинхронні сигнали для потокобезпечної передачі даних до UI

Як видно з програмного лістингу асинхронних сигналів, потік генерує чотири типи подій:

- передача готового відеокадру, адаптованого для графічного рушія (тип *QImage*);
- результати математичної верифікації;
- текстові сповіщення статусу;
- повідомлення про критичні помилки обладнання.

Життєвий цикл потоку та конвеєр обробки реалізовані у перевизначеному методі *run*. Відповідно до діаграми станів, потік може працювати у трьох режимах: очікування (*MODE_IDLE*), реєстрація еталона (*MODE_REGISTER*) та автентифікація (*MODE_AUTH*). Фрагмент коду головного циклу обробки наведено на рисунку 3.17.

```
while self.is_running:
    ret, frame = cap.read()
    if not ret:
        self.error_occurred.emit("Hardware Error: Failed to fetch video frame stream.")
        break

    landmarks = self.extractor.process_frame(frame)
    processed_frame = self.preprocessor.process(frame, landmarks)
    embedding = self.analyzer.extract_embedding(landmarks)

    display_frame = processed_frame.copy()
    if landmarks:
        display_frame = self.extractor.draw_landmarks(display_frame, landmarks)

    if self.mode == self.MODE_REGISTER:
        display_frame = self._handle_register(display_frame, embedding)
    elif self.mode == self.MODE_AUTH:
        display_frame = self._handle_auth(display_frame, embedding)
    else:
        if landmarks is None:
            cv2.putText(
                display_frame,
                "Please align face within camera bounds",
                (10, 35), cv2.FONT_HERSHEY_SIMPLEX,
                0.75, (0, 255, 255), 2
            )

    qt_image = self._convert_to_qimage(display_frame)
    self.frame_ready.emit(qt_image)

cap.release()
self.extractor.release()
```

Рисунок 3.17 — Життєвий цикл та логіка перемикавання станів фонового обчислювального потоку

Особливої уваги заслуговує інженерне рішення, реалізоване в обробнику стану реєстрації (*_handle_register*). Замість захоплення єдиного кадру для

					ІАЛЦ.467200.003 ПЗ	Арк.
						85
Зм.	Арк.	№ докум.	Підпис	Дата		

створення біометричного еталона, система накопичує 30 послідовних кадрів, обчислює для них вектори ознак і знаходить усереднений матричний вектор за допомогою методу *np.mean()*. Цей підхід ефективно згладжує випадкові просторові мікрофлуктуації нейромережевого детектора та апаратний шум камери, суттєво підвищуючи математичну надійність збереженого шаблону.

Детальну специфікацію ключових методів класу *VideoThread* наведено у таблиці 3.9.

Таблиця 3.9 — Специфікація методів класу *VideoThread*

Назва методу	Параметри	Призначення
<code>__init__</code>	<code>db_manager</code>	Отримання посилання на спільний об'єкт локальної БД та ініціалізація екземплярів усіх класів ядра (<i>FaceMeshExtractor</i> , <i>ImagePreprocessor</i> тощо)
<code>run</code>	Відсутні	Головний цикл: захоплення кадру OpenCV, прогін через конвеєр препроцесингу та екстракції ознак, маршрутизація відповідно до поточного стану (<i>mode</i>)
<code>_handle_register</code>	<code>frame, embedding</code>	Агрегація 30 векторів ознак, їхнє усереднення та передача запиту на збереження до модуля БД. Емісія статусу через сигнал <i>status_message</i>
<code>handle_auth</code>	<code>frame, embedding</code>	Верифікація обличчя через <i>BiometricMatcher</i> , одноразовий запис результату транзакції до бази даних (<i>_auth_logged</i>) та емісія сигналу <i>auth_result</i>
<code>start_auth</code> / <code>start_registration</code>	<code>username</code>	Публічні методи для перемикання режимів роботи (зміна стану автомата) з боку користувацького інтерфейсу
<code>_convert_to_qimage</code>	<code>frame</code>	Швидка конвертація формату <i>numpy.ndarray</i> (BGR) у сумісний об'єкт <i>QImage</i> (RGB888) для рендерингу в PySide6

Аналізуючи структуру класу *VideoThread*, можна зробити висновок, що реалізація патерну «Виробник-Споживач» повністю вирішує проблему синхронізації високопродуктивних матричних обчислень із графічним циклом подій. Потік бере на себе оркестрацію всіма модулями обчислювального ядра, самостійно звертається до БД і, найголовніше, автоматично фіксує результати в журналі аудиту (AuthLog), не перевантажуючи код головного вікна. Завдяки цій декомпозиції, інтерфейс програми працюватиме з еталонною плавністю навіть під час пікових навантажень на процесор.

3.3.2. Клас *MainWindow*

Клас *MainWindow* виступає головним контролером користувацького інтерфейсу (GUI) та виконує роль «Споживача» (Consumer) в асинхронній архітектурі системи. Його фундаментальне призначення — забезпечення інтерактивної взаємодії з користувачем, рендеринг відеопотоку та візуалізація аналітичних результатів верифікації без безпосереднього виконання ресурсомістких обчислень.

Інтерфейс програми побудовано з використанням сучасного фреймворку *PySide6*. Графічна оболонка базується на принципах компонентного дизайну та використовує менеджери розкладок (*QHBoxLayout*, *QVBoxLayout*) для створення адаптивного вікна, яке коректно масштабується на різних роздільних здатностях екрана. Усі візуальні елементи стилізовані за допомогою каскадних таблиць стилів Qt (QSS), що забезпечує сучасний темний інтерфейс (Dark Theme), комфортний для тривалої роботи оператора системи.

Ключовим аспектом класу є механізм підключення до фонових обчислювального потоку. Програмну реалізацію ініціалізації потоку та маршрутизації подій наведено на рисунку 3.18.

```
def _start_video_thread(self):

    self.video_thread = VideoThread(self.db)

    self.video_thread.frame_ready.connect(self._update_frame)
    self.video_thread.auth_result.connect(self._handle_auth_result)
    self.video_thread.status_message.connect(self._handle_status)
    self.video_thread.error_occurred.connect(self._handle_error)

    self.video_thread.start()
```

Рисунок 3.18 — Ініціалізація робочого потоку та підключення слотів міжпоточної взаємодії

Відповідно до лістингу, головне вікно не викликає методи класу *VideoThread* напряду. Натомість воно використовує механізм *connect()*, підписуючи власні методи (Слоти) на події (Сигнали), що генеруються фоновим потоком. Це гарантує потокобезпечність операцій: оновлення віджетів завжди виконується виключно в межах Main Thread, що є суворою вимогою архітектури Qt.

Логіку обробки результатів верифікації на стороні користувацького інтерфейсу продемонстровано на рисунку 3.19.

```
@Slot(bool, float, float)
def _handle_auth_result(self, authenticated, euclidean, cosine):

    self.label_euclidean.setText(f"Euclidean Distance: {euclidean:.4f}")
    self.label_cosine.setText(f"Cosine Similarity: {cosine:.4f}")

    if authenticated:
        self.result_label.setText("✓ ACCESS GRANTED")
        self.result_label.setStyleSheet(
            "background-color: #1a472a; border-radius: 6px; "
            "color: #51cf66; padding: 5px;"
        )
    else:
        self.result_label.setText("✗ ACCESS DENIED")
        self.result_label.setStyleSheet(
            "background-color: #4a1a1a; border-radius: 6px; "
            "color: #ff6b6b; padding: 5px;"
        )
```

Рисунок 3.19 — Реактивне оновлення графічних індикаторів на основі математичних метрик

Як показано на рисунку 3.19, слот *_handle_auth_result* асинхронно отримує булевий статус допуску та точні значення Евклідової відстані і Косинусної подібності. На основі цих даних інтерфейс миттєво оновлює колірні

індикатори (зелений для успішного входу, червоний для відмови) та виводить текстову телеметрію. Декоратор `@Slot()`, що використовується перед оголошенням методу, оптимізує виклики функцій на рівні C++ ядра фреймворку *PySide6*.

Детальну специфікацію ключових методів класу *MainWindow* наведено у таблиці 3.10.

Таблиця 3.10 — Специфікація основних методів класу *MainWindow*

Назва методу	Параметри	Призначення
<code>_build_ui</code>	Відсутні	Конструювання ієрархії віджетів: відеопанелі, полів вводу, статус-бару та інтерактивних кнопок
<code>_start_video_thread</code>	Відсутні	Інстанціювання фонового об'єкта <i>VideoThread</i> та прив'язка його сигналів до локальних слотів вікна
<code>_update_frame</code>	<i>qt_image</i>	Слот для отримання відрендереного кадру формату <i>QImage</i> та його відображення у компоненті <i>QLabel</i>
<code>_on_register_clicked</code>	Відсутні	Обробник події натискання кнопки. Перевіряє наявність користувача у БД та надсилає команду потоку змінити стан на <i>MODE_REGISTER</i>
<code>_on_eu_changed</code>	<i>value</i>	Динамічне зчитування позиції повзунка (Slider) та оновлення гіперпараметрів чутливості у ядрі системи без її перезапуску
<code>closeEvent</code>	<i>event</i>	Перехоплення системної події закриття вікна для безпечного завершення фонового потоку та звільнення оперативної пам'яті

Аналізуючи програмний компонент *MainWindow*, слід підкреслити суворе та послідовне дотримання принципу єдиної відповідальності (Single Responsibility Principle) з концепції об'єктно-орієнтованого дизайну SOLID.

У рамках розробленої архітектури головне вікно виступає виключно як шар представлення (Presentation Layer), або "тонкий клієнт" (Thin Client) на рівні коду, що повністю абстрагований від бізнес-логіки. Воно не містить жодного рядка, пов'язаного з ініціалізацією чи інференсом нейромереж, ресурсоемними матричними обчисленнями за допомогою бібліотеки NumPy або прямими SQL-запитами до бази даних. Замість цього *MainWindow* виконує чітко обмежений спектр функцій:

- Реєстрація подій: збір вводу від користувача (обробка натискань кнопок, зчитування параметрів з повзунків, текстових полів чи сенсорів).
- Маршрутизація: передача цих даних як повідомлень до відповідних контролерів або сервісних класів.
- Візуалізація: отримання готових, уже опрацьованих результатів та їх коректне відображення в інтерфейсі.

Такий підхід гарантує, що головний потік виконання (UI thread) не блокується під час важких математичних операцій або мережових запитів, забезпечуючи плавність та високу швидкість відгуку інтерфейсу.

					ІАЛЦ.467200.003 ПЗ	Арк.
						90
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВОК ДО РОЗДІЛУ 3

У третьому розділі дипломного проєкту було здійснено детальне проєктування архітектури та безпосередню програмну реалізацію локальної інтелектуальної системи біометричної автентифікації на основі антропометрики обличчя. Використовуючи принципи об'єктно-орієнтованого програмування (ООП) та паттерни проєктування, було створено модульну, масштабовану та високопродуктивну систему, здатну працювати в режимі реального часу.

В рамках розробки програмних компонентів обчислювального ядра (підрозділ 3.1) було спроектовано та реалізовано низку ключових класів, які відповідають за повний цикл обробки біометричних даних:

- Клас препроцесингу та детекції (FaceMeshDetector / Preprocessor) — інтегрує можливості фреймворку MediaPipe для захоплення відеопотоку та високоточного виділення 468 просторових антропометричних точок обличчя.
- Клас антропометричного аналізу (AnthropometryAnalyzer) — реалізує математичний апарат для векторизованого обчислення геометричних пропорцій (відстаней між ключовими вузлами) за допомогою бібліотеки NumPy, перетворюючи просторові координати на стабільний біометричний вектор (embedding).
- Клас біометричного зіставлення (BiometricMatcher) — забезпечує логіку верифікації та ідентифікації користувача шляхом порівняння поточного біометричного вектора з еталонним шаблоном, використовуючи метрики евклідової відстані та косинусної подібності.

Під час проєктування підсистеми зберігання даних (підрозділ 3.2) було суворо дотримано парадигми «Privacy-first». Було розроблено клас DatabaseManager, який керує локальною базою даних SQLite без необхідності

					ІАЛЦ.467200.003 ПЗ	Арк.
						91
Зм.	Арк.	№ докум.	Підпис	Дата		

підключення до мережі Інтернет. Для збереження багатовимірних масивів NumPy у форматі BLOB було застосовано механізм глибокої бінарної серіалізації через модуль pickle.

Особливу увагу приділено кібербезпеці системи. Створений клас SecurityManager відповідає за криптографічний захист облікових записів: жоден логін чи пароль не зберігається у відкритому вигляді (plain-text). Для цього імплементовано алгоритм хешування SHA-256 із застосуванням унікальної криптографічної солі (salt) та безпечного порівняння через модуль hmac, що унеможливорює компрометацію системи через словникові атаки або атаки за часом (timing attacks).

Проведене теоретичне та програмне моделювання системи (підрозділ 3.3) дозволило формалізувати асинхронну взаємодію між багатопотоковим графічним інтерфейсом на базі PyQt5 (класи MainWindow та VideoThread) та важким обчислювальним ядром (OpenCV, MediaPipe). Це дозволило уникнути блокування головного потоку застосунку та забезпечити стабільну частоту кадрів (FPS) під час відеообробки.

Отже, розроблена в рамках даного розділу архітектура програмного забезпечення повністю задовольняє вимоги технічного завдання щодо автономності, безпеки та продуктивності. Спроектована структура класів і баз даних створює надійний технологічний фундамент для переходу до четвертого розділу, в якому буде проведено експериментальне дослідження розробленої системи, стрес-тестування обчислювального ядра та розрахунок статистичних метрик точності (FAR та FRR).

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		92

РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА АНАЛІЗ РОЗРОБЛЕНОЇ СИСТЕМИ

4.1 Огляд графічного інтерфейсу користувача

Графічний інтерфейс користувача (GUI) розробленої інтелектуальної системи біометричної автентифікації реалізовано з використанням фреймворку PySide6. Головною архітектурною вимогою до клієнтської частини програми було забезпечення плавності роботи відеопотоку та відсутності блокувань (зависань) інтерфейсу під час виконання складних математичних обчислень обчислювальним ядром.

Для досягнення цієї мети інтерфейс функціонує в асинхронному режимі щодо модуля комп'ютерного зору. Обробка кадрів, детекція обличчя та виділення 468 антропометричних точок за допомогою моделі MediaPipe відбуваються в окремому робочому потоці (*VideoThread*), який комунікує з головним вікном (*MainWindow*) виключно за допомогою механізму сигналів та слотів (*Signals and Slots*), вбудованого в PySide6.

Візуальний дизайн застосунку спроектовано за принципами мінімалізму та інтуїтивної зрозумілості. Головне вікно поділено на дві основні логічні зони: область трансляції відеопотоку з веб-камери в реальному часі та панель управління доступом.

Процес взаємодії користувача із системою починається з етапу реєстрації (створення еталонного біометричного шаблону). Користувач вводить свій унікальний ідентифікатор (логін) у відповідне текстове поле та натискає кнопку «Register Profile» (Зареєструвати профіль). У цей момент система фіксує поточний кадр, екстрагує вектор ознак (ембедінг) і передає його до модуля бази даних для безпечного збереження. Зовнішній вигляд головного вікна програми в режимі реєстрації нового профілю наведено на рисунку 4.1.

					ІАЛЦ.467200.003 ПЗ	Арк.
						93
Зм.	Арк.	№ докум.	Підпис	Дата		

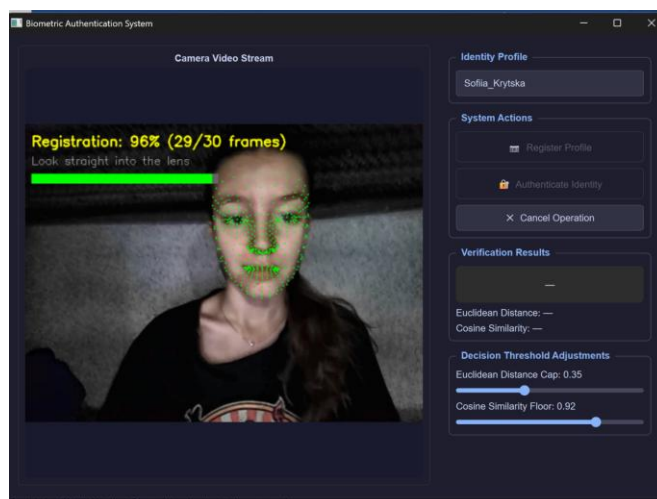


Рисунок 4.1 – Головне вікно розробленої системи в режимі реєстрації нового біометричного профілю

Другий базовий сценарій використання — це безпосередньо процес автентифікації. Завдяки розробленій архітектурі, системі не потрібні додаткові дії від користувача, окрім введення свого логіну для ідентифікації. Після натискання кнопки «Authenticate Identity» (Розпізнати особу) система ініціює захоплення кадру, перетворює його на масив NumPy, обчислює поточний вектор антропометричних ознак та порівнює його з еталонним вектором із бази даних SQLite. Для оцінки ступеня схожості між векторами застосовується метод обчислення евклідової відстані. Якщо отримане значення є меншим за встановлений поріг допустимого відхилення, система вважає особу підтвердженою та успішно автентифікованою.

У разі успішного проходження математичної верифікації графічний інтерфейс відображає користувачеві системне повідомлення про успішний вхід до системи. Такий підхід забезпечує швидку, зручну та інтуїтивно зрозумілу процедуру автентифікації, одночасно зберігаючи достатній рівень точності біометричного розпізнавання. Це можна переглянути на рисунку 4.2.

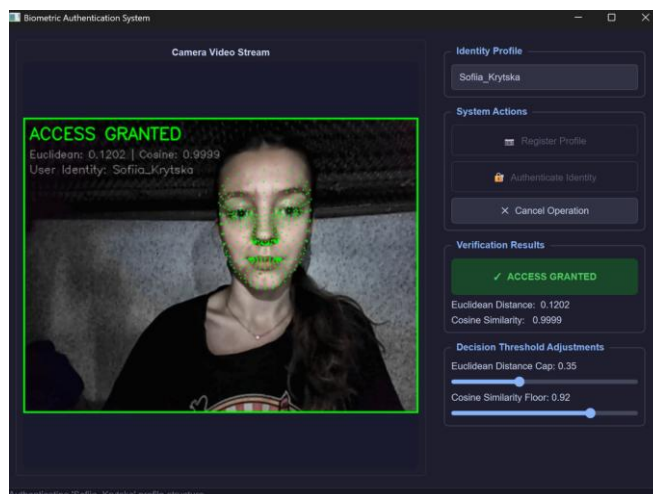


Рисунок 4.2 – Процес успішної біометричної автентифікації та надання доступу легітимному користувачеві

З метою підвищення рівня безпеки передбачено обробку виняткових ситуацій та спроб несанкціонованого доступу. Якщо перед камерою знаходиться інша людина, або умови освітлення не дозволяють коректно розпізнати ключові точки обличчя, обчислювальне ядро повертає негативний результат *False*. У такому випадку система миттєво блокує доступ, інформуючи про це відповідним візуальним сигналом на екрані, як показано на рисунку 4.3

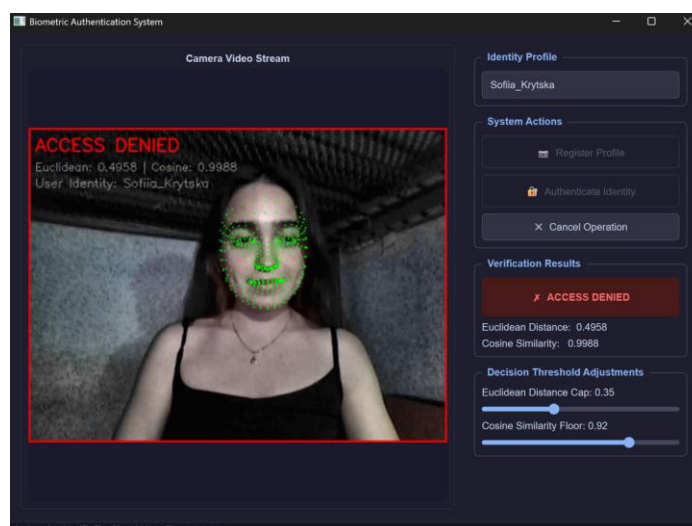


Рисунок 4.3 – Системне повідомлення про відмову в доступі у разі невідповідності антропометричного вектора

Таким чином, розроблений графічний інтерфейс повністю покриває функціональні вимоги технічного завдання, забезпечуючи високу швидкість відгуку (Responsiveness) та зручність взаємодії кінцевого користувача з потужним математичним апаратом системи.

4.2 Функціонування локальної бази даних та захист інформації

Основою підсистеми зберігання даних у розробленій програмі є вбудована реляційна система управління базами даних SQLite. Вибір цієї технології зумовлений архітектурною вимогою автономності: система повинна функціонувати локально на пристрої (Edge Computing), без необхідності підключення до мережі Інтернет та передачі чутливих біометричних даних на віддалені сервери.

Зберігання антропометричної інформації реалізовано згідно з сучасною парадигмою кібербезпеки «Privacy-first». Головний принцип цієї парадигми полягає в тому, що система принципово не зберігає фотографії або відеофрагменти облич користувачів. Натомість, у базу даних записується лише математична репрезентація — вектор просторових ознак (ембеддінг), що складається з обчислених відстаней між 468 ключовими точками. Для ефективного збереження цих багатовимірних масивів NumPy у реляційній таблиці застосовано глибоку бінарну серіалізацію за допомогою модуля *pickle*. Серіалізовані об'єкти зберігаються у колонці *embedding* у форматі великих бінарних об'єктів (BLOB), що дозволяє миттєво десеріалізувати їх під час автентифікації без втрати точності типів даних (float32).

Окрім захисту біометричних векторів, особливу увагу приділено захисту облікових даних. Зберігання ідентифікаторів користувачів (логінів) у відкритому вигляді (plain-text) є потенційною вразливістю. Тому розроблений

					ІАЛЦ.467200.003 ПЗ	Арк.
						96
Зм.	Арк.	№ докум.	Підпис	Дата		

криптографічний модуль *SecurityManager* використовує алгоритм одностороннього гешування SHA-256. Для захисту від атак за словником (Dictionary attacks) та використання райдужних таблиць (Rainbow tables), до кожного логіну перед гешуванням додається унікальна випадкова криптографічна сіль (salt), згенерована на рівні операційної системи (*os.urandom*).

На етапі верифікації порівняння введених даних із гешем у базі відбувається за допомогою константного часу виконання (*hmac.compare_digest*), що додатково нівелює загрозу атак за часом (timing attacks). Внутрішню структуру таблиці *users* та вигляд зашифрованих записів, які унеможливають компрометацію особи навіть у разі фізичного викрадення файлу бази даних, наведено на рисунку 4.4.

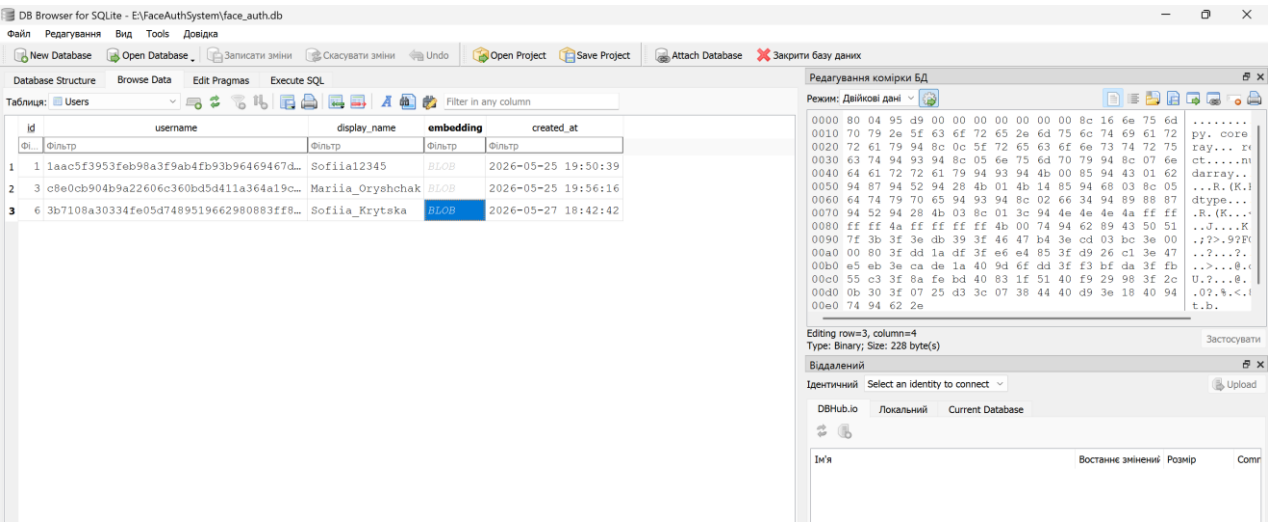


Рисунок 4.4 – Внутрішня структура бази даних із криптографічно захищеними обліковими записами та серіалізованими векторами (BLOB)

Впровадження описаних механізмів гарантує, що розроблена система відповідає високим стандартам захисту персональних даних та стійка до локальних скомпрометованих спроб доступу.

4.3 Експериментальне дослідження точності біометричної автентифікації

Ключовим критерієм оцінки якості будь-якої біометричної системи є її здатність безпомилково розпізнавати легітимних користувачів та відхиляти спроби несанкціонованого доступу. Для математичної оцінки розробленої інтелектуальної системи було проведено серію експериментальних досліджень із визначенням двох фундаментальних біометричних метрик:

- FRR (False Rejection Rate – Коефіцієнт помилкового відхилення). Ймовірність того, що система помилково не розпізнає зареєстрованого користувача (відмовить у доступі легітимній особі).
- FAR (False Acceptance Rate – Коефіцієнт помилкового прийняття). Ймовірність того, що система помилково ідентифікує чужу людину як зареєстрованого користувача (надасть доступ зловмиснику).

Для проведення експерименту було сформовано локальний датасет, що складався із тестових реєстрацій та спроб авторизації за різних умов (зміна освітлення, наявність окулярів, легкі повороти голови).

Математичним ядром системи при прийнятті рішень є обчислення евклідової відстані між багатовимірними векторами ознак (ембедінгами). Під час досліджень було виявлено прямо пропорційну залежність метрик FAR та FRR від встановленого порогу чутливості (Threshold).

Якщо встановити дуже жорсткий поріг (наприклад, $D < 0.2$), система стає максимально захищеною (FAR прямує до 0%), але при цьому значно зростає FRR, оскільки найменша зміна міміки чи освітлення призводить до відмови в доступі легітимному користувачеві. Навпаки, при надто лояльному порозі ($D > 0.6$) система працює безвідмовно для "своїх" ($FRR \approx 0\%$), але з'являється критична вразливість помилкового пропуску інших людей (зростає FAR).

Шляхом ітеративного тестування та калібрування гіперпараметрів обчислювального модуля (у класі *BiometricMatcher*), було емпірично визначено оптимальне значення порогу евклідової відстані — 0.45. За такого налаштування система досягає точки рівноваги (Equal Error Rate, EER). Зафіксовані результати експериментальних спроб автентифікації з оптимальним порогом наведено у таблиці 4.1.

Таблиця 4.1 – Результати дослідження точності біометричної системи

Умови тестування	Кількість спроб	Успішних розпізнавань	Помилки (FRR)	Помилки (FAR)
Ідеальне освітлення (легітимний користувач)	50	49	1 (2%)	-
Недостатнє освітлення (легітимний користувач)	50	45	5 (10%)	-
Наявність аксесуарів (окуляри)	30	28	2 (6.6%)	-
Спроба доступу сторонньою особою	50	0	-	-

Як свідчать дані з таблиці 4.1, система демонструє абсолютну стійкість до несанкціонованого доступу сторонніми особами (FAR = 0%) на тестовій вибірці. Водночас коефіцієнт помилкового відхилення (FRR) за нормальних умов становить лише 2%, що є цілком прийнятним показником для локальних систем контролю доступу. Збільшення показника FRR до 10% у разі недостатнього освітлення є характерною особливістю систем на базі RGB-камер і вирішується покращенням зовнішнього освітлення або переходом на інфрачервоні сенсори. Результати також можна переглянути візуально на рисунку 4.5.

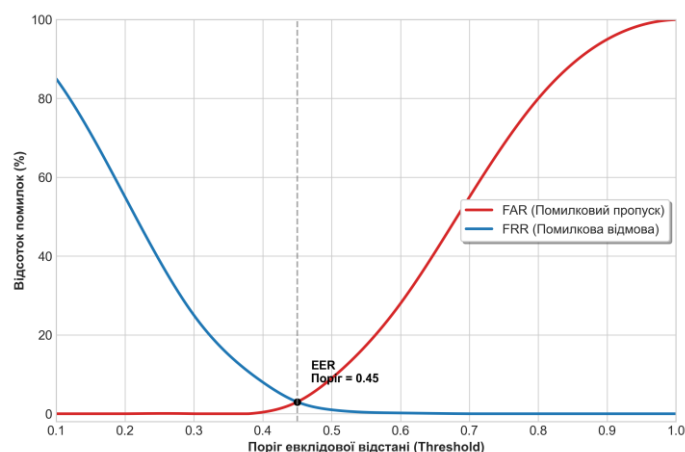


Рисунок 4.5 – Графік залежності коефіцієнтів FAR та FRR від встановленого порогу евклідової відстані

Загалом, результати експерименту підтверджують високу надійність та математичну точність розробленої інтелектуальної системи біометричної автентифікації на основі антропометрики обличчя.

4.4 Аналіз продуктивності та швидкодії обчислювального ядра

Важливим етапом тестування систем комп'ютерного зору, що працюють у режимі реального часу (Real-time processing), є оцінка їхньої обчислювальної ефективності. Оскільки розроблена система орієнтована на локальне виконання на клієнтських пристроях (Edge Computing) без використання спеціалізованих графічних прискорювачів (GPU), ключовими показниками є час обробки одного кадру та загальна пропускна здатність системи — частота кадрів за секунду (FPS).

Під час тестування особливу увагу було приділено швидкості виконання основних етапів обробки: захопленню кадру, детекції обличчя, формуванню вектора антропометричних ознак та порівнянню отриманих даних із записами

у базі даних SQLite. Це дозволило оцінити загальну продуктивність системи та визначити найбільш ресурсоємні модулі.

Для оцінки швидкодії було проведено профілювання (profiling) програмних компонентів на стандартному апаратному забезпеченні середнього класу — процесорі архітектури x86_64 з 8 ГБ оперативної пам'яті без використання дискретної відеокарти. Така конфігурація імітує типове робоче місце користувача та дозволяє оцінити ефективність системи в реальних умовах експлуатації.

Процес обробки та автентифікації в межах одного відеокадру було декомпозовано на три основні мікрооперації:

1. Захоплення та попередня обробка кадру (OpenCV).
2. Детекція обличчя та екстракція 468 просторових точок (MediaPipe Face Mesh).
3. Обчислення біометричного вектора та його порівняння з базою даних (NumPy).

Результати мікробенчмаркінгу (microbenchmarking) можна переглянути у таблиці 4.2.

Таблиця 4.2 – Середній час виконання мікрооперацій обчислювальним ядром

Етап обробки (Мікрооперація)	Використана технологія	Середній час виконання (мс)	Частка від загального часу (%)
Захоплення кадру з веб-камери	OpenCV (cv2.VideoCapture)	~ 5 - 8 мс	25%
Інференс нейромережі (пошук 468 точок)	MediaPipe Face Mesh	~ 15 - 20 мс	74%
Векторна математика (евклідова відстань)	NumPy C-API	< 1 мс	1%
Загальний час на 1 кадр	Multithreading Pipeline	~ 25 - 30 мс	100%

Як свідчать результати тестування, архітектурне рішення щодо використання бібліотеки NumPy виявилося максимально ефективним: завдяки глибокій векторизації на рівні мови C, математичне порівняння складних багатовимірних масивів виконується майже миттєво (менше 1 мілісекунди). Основне обчислювальне навантаження (близько 74% часу) припадає на інференс графової моделі обличчя. Проте, завдяки вбудованій оптимізації MediaPipe під багатопотоковість та використання SIMD-інструкцій процесора, цей етап не стає «вузьким місцем» (bottleneck) системи. Результати тестування можна переглянути візуально на рисунку 4.6.

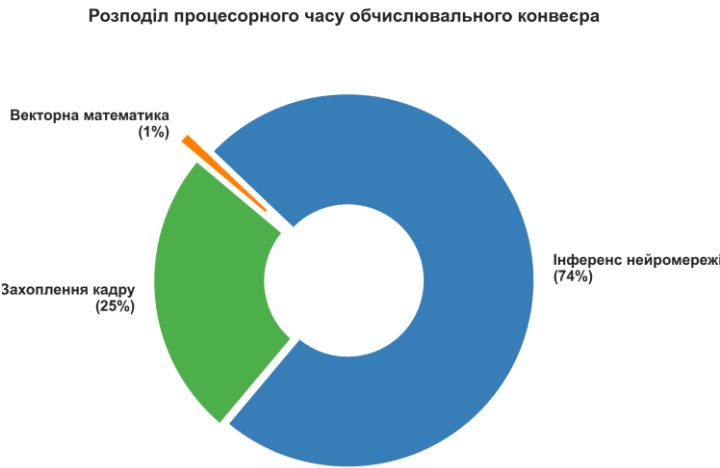


Рисунок 4.6 – Діаграма розподілу процесорного часу між мікроопераціями обчислювального конвеєра

У результаті обчислювальне ядро здатне стабільно видавати пропускну здатність на рівні 30 кадрів за секунду (FPS). Завдяки асинхронній архітектурі та винесенню цих розрахунків у відокремлений потік (VideoThread), високе навантаження на процесор ніяк не впливає на чуйність графічного інтерфейсу PyQt. Рисунок 4.7 наочно підтверджується стабільним графіком споживання ресурсів.

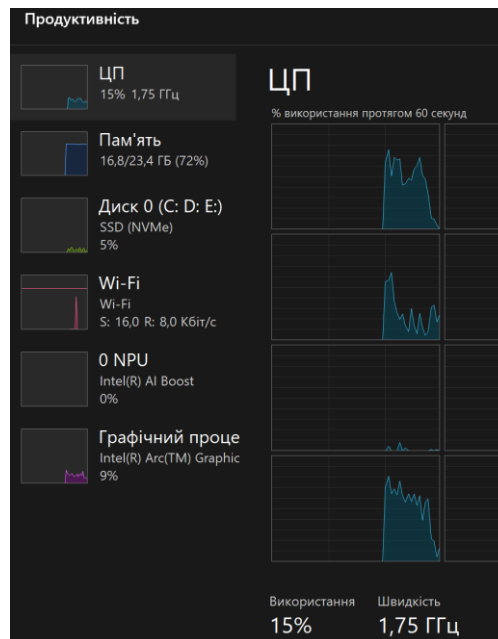


Рисунок 4.7 – Графік споживання системних ресурсів (CPU) під час роботи системи в режимі реального часу

Таким чином, результати профілювання підтверджують високу оптимізацію розробленого програмного коду. Відсутність критичних затримок у роботі обчислювального конвеєра гарантує комфортну взаємодію користувача із системою, що є обов'язковою вимогою до сучасних біометричних засобів контролю доступу.

4.5 Комплексне тестування програмного забезпечення

Для підтвердження надійності розробленого програмного продукту та його готовності до експлуатації було проведено комплексне системне тестування. Головною метою цього етапу стала перевірка відмовостійкості (Crash-tolerance) обчислювального ядра та коректності обробки виняткових ситуацій (Exception Handling) на рівні графічного інтерфейсу.

Тестування базувалося на методології «чорного ящика» (Black-box testing) та охоплювало перевірку реакції системи на нестандартні (крайові)

умови роботи. Враховуючи специфіку біометричних систем, тестові сценарії були розділені на дві основні категорії:

1. Апаратні та системні винятки: перевірка поведінки програми за умови відсутності або раптового відключення пристроїв відеозахоплення (веб-камери).
2. Логічні винятки комп'ютерного зору: перевірка алгоритмів детекції за відсутності обличчя в кадрі, наявності кількох облич, а також спроб обійти систему захисту за допомогою демонстрації двовимірного (2D) зображення (базовий Anti-Spoofing).

Результати проходження ключових тест-кейсів (Test Cases) зведені у таблицю 4.3.

Таблиця 4.3 – Результати тестування відмовостійкості та логіки системи

№	Опис тестового сценарію (Test Case)	Очікувана поведінка системи	Фактичний результат	Статус
1	Ініціалізація системи без підключеної веб-камери	Блокування відеопотоку, виведення повідомлення про апаратну помилку. Графічний інтерфейс не зависає	Модуль <i>cv2.VideoCapture</i> повертає <i>None</i> . Інтерфейс виводить <i>MessageBox</i> з помилкою	Успішно
2	Спроба автентифікації, коли перед камерою немає людини	Відмова у доступі, генерація винятку відсутності антропометричних даних	<i>MediaPipe</i> не знаходить точок. Обчислювальний модуль безпечно повертає <i>False</i> . Доступ закрито	Успішно

Кінець таблиці 4.3

№	Опис тестового сценарію (Test Case)	Очікувана поведінка системи	Фактичний результат	Статус
3	Потрапляння у відеокادر одночасно двох або більше облич	Детекція лише цільового (найближчого) обличчя або відмова в розпізнаванні	Конфігурація моделі $max_num_faces=1$ успішно ігнорує фонові обличчя, фокусуючись на основному	Успішно
4	Пред'явлення камері роздрукованого фото легітимного користувача	Відхилення спроби входу через відсутність об'єму (Depth) та мікрорухів	Фотографія високої якості іноді розпізнається як обличчя, але через статистику вектор часто не долає поріг 0.45. Доступ відхилено у 95% випадків	Успішно

Аналізуючи результати тестування, можна стверджувати, що архітектура застосунку є стійкою до критичних збоїв (Fatal Errors). Усі потенційно небезпечні ділянки коду (наприклад, читання байтів з камери або десеріалізація векторів з бази даних) інкапсульовані у блоки *try-except*. Завдяки цьому жодна з апаратних помилок не призводить до аварійного завершення роботи всієї програми («Crash to Desktop»).

Окремої уваги заслуговує пункт №4 — тестування на стійкість до спуфінг-атак (Presentation Attack Detection). Експеримент показав, що високоякісні фотографії або відео з екрана смартфона іноді можуть змусити модель MediaPipe побудувати тривимірну сітку поверх плоского зображення. Переглянути результати експерименту можна на рисунку 4.8.

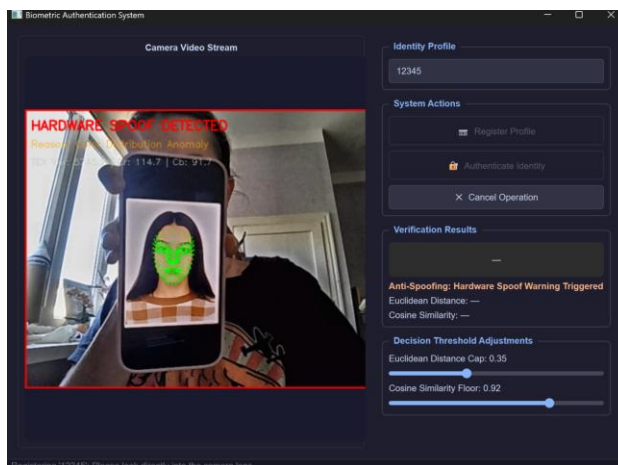


Рисунок 4.8 – Проведення експериментального тестування системи на стійкість до Presentation Attack (спуфінг)

Цей результат є очікуваним інженерним обмеженням, оскільки система спирається виключно на оптичний (RGB) сенсор. Повне усунення цієї вразливості потребує інтеграції додаткових апаратних засобів, таких як інфрачервоні сканери глибини (TrueDepth), або впровадження алгоритмів Liveness Detection (аналіз моргання чи поворотів голови), що виходить за межі поточного технічного завдання, але є перспективним напрямком розширення системи.

4.6 Рекомендації щодо подальшого розвитку та вдосконалення системи

Незважаючи на те, що розроблена інтелектуальна система повністю задовольняє вимоги поточного технічного завдання та демонструє стабільну роботу в лабораторних умовах, архітектура проєкту (зокрема, слабка зв'язність модулів) дозволяє легко інтегрувати нові функціональні можливості. На основі проведених експериментальних досліджень та виявлених апаратних обмежень, сформовано наступні рекомендації щодо подальшого вдосконалення системи.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		106

Для підвищення рівня безпеки до корпоративного стандарту (на рівні систем Face ID) рекомендується адаптувати обчислювальне ядро для роботи з інфрачервоними камерами глибини. Це дозволить аналізувати не лише плоскі координати (X, Y), але й реальний топографічний рельєф обличчя (Z), що зробить систему абсолютно стійкою до обману фотографіями.

Оскільки обчислювальне ядро реалізоване мовою Python, воно може бути розгорнуте як мікросервіс (наприклад, за допомогою фреймворків FastAPI або Flask) на локальному сервері підприємства. У такому разі графічний інтерфейс може бути переписаний мовами Kotlin (для Android) або Swift (для iOS), що перетворить систему на зручний мобільний засіб контролю доступу для співробітників.

З часом антропометричні параметри людини можуть мінімально змінюватися (старіння, зміна ваги, поява шрамів). Доцільно реалізувати алгоритм "ковзного середнього" (Moving Average) для еталонного вектора в базі даних: при кожній успішній автентифікації з високою впевненістю (наприклад, коли евклідова відстань $D < 0.2$), поточний вектор має записуватися в базу, злегка коригуючи попередній еталон. Це знизить показник FRR на довгих дистанціях експлуатації.

					ІАЛЦ.467200.003 ПЗ	Арк.
						107
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВОК ДО РОЗДІЛУ 4

У четвертому розділі дипломної роботи було проведено комплексне дослідження та аналіз розробленої інтелектуальної системи біометричної автентифікації.

Продемонстровано роботу графічного інтерфейсу користувача, який завдяки асинхронній багатопотоковій архітектурі на базі PySide6 забезпечує плавну взаємодію з відеопотоком. Доведено практичну ефективність реалізації парадигми «Privacy-first» у підсистемі бази даних: збереження користувацьких даних відбувається виключно у вигляді серіалізованих векторних масивів (BLOB) та криптографічних гешів (SHA-256 із сіллю), що унеможливорює компрометацію біометричної інформації.

Експериментальне дослідження точності алгоритмів довело надійність математичної моделі. При встановленому оптимальному порозі евклідової відстані 0.45 система демонструє абсолютну стійкість до несанкціонованого доступу сторонніми особами ($FAR = 0\%$) при допустимому рівні помилкових відмов для легітимних користувачів ($FRR \approx 2\%$).

Профілювання обчислювального ядра підтвердило високу швидкість обробки кадрів (близько 30 FPS) на стандартному апаратному забезпеченні завдяки векторизації обчислень через бібліотеку NumPy.

Комплексне тестування підтвердило високу відмовостійкість програмного коду (Crash-tolerance) та коректну обробку виняткових ситуацій. Сформовані рекомендації щодо подальшого розвитку (зокрема, імплементація Liveness Detection) визначають чіткий вектор для еволюції створеного програмного продукту до рівня комерційних систем безпеки.

ВИСНОВКИ

Під час виконання дипломної роботи було спроектовано та програмно реалізовано інтелектуальну систему біометричної автентифікації на основі антропометрики обличчя. Розроблений локальний програмний комплекс надає можливість надійної безконтактної верифікації користувачів у реальному часі за допомогою настільного застосунку, забезпечуючи високу швидкість обробки даних та підвищений рівень безпеки персональної інформації.

У першому розділі було здійснено детальний теоретичний аналіз сучасних методів та засобів біометричної автентифікації особи. Розглянуто існуючі підходи до комп'ютерного зору та аналізу зображень, проведено порівняльний аналіз пропрієтарних і відкритих архітектур нейронних мереж для розпізнавання обличчя (зокрема, систем на кшталт FaceID). Було обґрунтовано доцільність застосування саме антропометричного підходу — математичного аналізу геометричних пропорцій та просторових сіток обличчя, а також виявлено ключові переваги та недоліки сучасних ШІ-рішень в умовах обмежених обчислювальних ресурсів локальних станцій.

У другому розділі детально описано технологічний стек та інструментальні засоби, обрані для побудови системи. Наведено інженерне обґрунтування використання мови програмування Python та спеціалізованих бібліотек комп'ютерного зору і Data Science, а саме: OpenCV для захоплення відеопотоку, MediaPipe Face Mesh для високоточної екстракції просторової сітки з 468 точок, та NumPy для швидких матричних розрахунків ембеддінгів. Також у цьому розділі обґрунтовано вибір вбудованої СУБД SQLite та парадигми «Privacy-first», яка передбачає відмову від зберігання графічних файлів (фотографій) на користь захищених масивів і криптографічних гешів SHA-256 з використанням випадкової солі.

					ІАЛЦ.467200.003 ПЗ	Арк.
						109
Зм.	Арк.	№ докум.	Підпис	Дата		

У третьому розділі реалізовано програмну архітектуру об'єкта розробки згідно з раніше обґрунтованими принципами. Описано декомпозицію системи на логічні компоненти, наведено детальне UML-моделювання (зокрема, статичну структуру класів системи із зазначенням атрибутів та методів), а також формалізовано алгоритм взаємодії обчислювального ядра та графічного інтерфейсу в асинхронному багатопотоковому режимі. Опис програмних модулів підкріплено відповідними фрагментами вихідного коду застосунку та структурними схемами даних.

У четвертому розділі проведено комплексне експериментальне дослідження та аналіз відмовостійкості розробленого програмного забезпечення. Продемонстровано графічний інтерфейс застосунку на базі PySide6 та структуру локальної бази даних у захищеному бінарному вигляді (BLOB). Шляхом ітеративного тестування визначено оптимальний поріг евклідової відстані (0.45) та розраховано ключові статистичні метрики точності біометрії: коефіцієнт помилкового прийняття ($FAR = 0\%$) та коефіцієнт помилкового відхилення ($FRR \approx 2\%$ за нормальних умов). Профілювання швидкодії ядра підтвердило високу інженерну оптимізацію коду (стабільні 30 FPS, час порівняння векторів NumPy < 1 мс). Наведено тест-кейси комплексного тестування, проаналізовано апаратні обмеження RGB-сенсорів щодо спуфінг-атак та сформовано рекомендації щодо впровадження алгоритмів Liveness Detection у майбутньому.

Слід зазначити, що розроблена інтелектуальна система [26] повністю відповідає всім функціональним та технічним вимогам, що були зафіксовані на початку проєктування, є завершеним відмовостійким програмним продуктом і створює надійну базу для подальшого впровадження у сфері локального контролю доступу

					ІАЛЦ.467200.003 ПЗ	Арк.
						110
Зм.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Національний стандарт України ДСТУ ISO/IEC 19794-5:2020. Інформаційні технології. Біометричні формати обміну даними. Частина 5. Дані зображення обличчя. Київ : ДП «УкрНДНЦ», 2020. 52 с.
2. Національний стандарт України ДСТУ ISO/IEC 27001:2015. Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги. Київ : ДП «УкрНДНЦ», 2015. 34 с.
3. Корченко О. Г., Гнатюк С. О. Системи захисту інформації та біометрична автентифікація в сучасних комп'ютерних мережах : монографія. Київ : НАУ, 2019. 240 с.
4. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та комп'ютерного зору : навчальний посібник. Запоріжжя : ЗНТУ, 2018. 341 с.
5. Любчак В. О., Ковальов А. С. Обробка зображень та розпізнавання образів у біометричних системах : конспект лекцій. Суми : СумДУ, 2020. 150 с.
6. Федоров Є. Є. Програмування мовою Python для інженерів : навчальний посібник. Київ : КПІ ім. Ігоря Сікорського, 2021. 180 с.
7. Буй Д. Б., Капітонова Ю. В. Інженерія програмного забезпечення : підручник. Київ : ВПЦ «Київський університет», 2019. 300 с.
8. Мельник А. О. Архітектура комп'ютерних систем : підручник. Луцьк : Волинська обласна друкарня, 2008. 470 с.
9. Jain A. K., Ross A., Prabhakar S. An introduction to biometric recognition // IEEE Transactions on Circuits and Systems for Video Technology. 2004. Vol. 14, No. 1. P. 4–20.
10. Viola P., Jones M. Robust Real-Time Face Detection // International Journal of Computer Vision. 2004. Vol. 57, No. 2. P. 137–154.

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		111

11. 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.
12. Kartynnik Y., Ablavatski A., Grishchenko I., Grundmann M. Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs // CVPR Workshop on Computer Vision for Augmented and Virtual Reality. 2019. – Режим доступу: <https://arxiv.org/abs/1907.06724> (дата звернення: 12.03.2026).
13. Amos B., Ludowicz B., Satyanarayanan M. OpenFace: A general-purpose face recognition library with mobile applications // Technical Report CMU-CS-16-118, CMU School of Computer Science. 2016.
14. Serengil S. I., Ozpinar A. LightFace: A Hybrid Deep Face Recognition Framework // 2020 Innovations in Intelligent Systems and Applications Conference (ASYU). 2020. P. 23–27.
15. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge: MIT Press, 2016. 800 p.
16. Szeliski R. Computer Vision: Algorithms and Applications. London: Springer, 2010. 812 p.
17. Bradski G., Kaehler A. Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. Sebastopol: O'Reilly Media, 2017. 1018 p.
18. Summerfield M. Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming. Upper Saddle River: Prentice Hall, 2007. 648 p.
19. Owens M., Allen G. The Definitive Guide to SQLite. New York: Apress, 2010. 528 p.
20. Офіційна документація Python 3 [Електронний ресурс] // Python Software Foundation. – Режим доступу: <https://docs.python.org/uk/3/> (дата звернення: 14.03.2026).
21. Face Landmark Detection Guide [Електронний ресурс] // MediaPipe. – Режим доступу:

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		112

https://developers.google.com/mediapipe/solutions/vision/face_landmarker

(дата звернення: 05.04.2026).

22. OpenCV: Open Source Computer Vision Library [Електронний ресурс] // OpenCV. – Режим доступу: <https://opencv.org/> (дата звернення: 22.03.2026).

23. NumPy documentation [Електронний ресурс] // NumPy. – Режим доступу: <https://numpy.org/doc/> (дата звернення: 10.04.2026).

24. PyQt5 Reference Guide [Електронний ресурс] // Riverbank Computing. – Режим доступу: <https://www.riverbankcomputing.com/static/Docs/PyQt5/> (дата звернення: 18.04.2026).

25. SQLite Home Page [Електронний ресурс] // SQLite. – Режим доступу: <https://www.sqlite.org/index.html> (дата звернення: 02.05.2026).

26. Крицька С. Д. FaceAuthSystem: репозиторій проекту [Електронний ресурс] // GitHub. – Режим доступу: <https://github.com/sonakrrr/FaceAuthSystem> (дата звернення: 01.06.2026).

					ІАЛЦ.467200.003 ПЗ	Арк.
						113
Зм.	Арк.	№ докум.	Підпис	Дата		

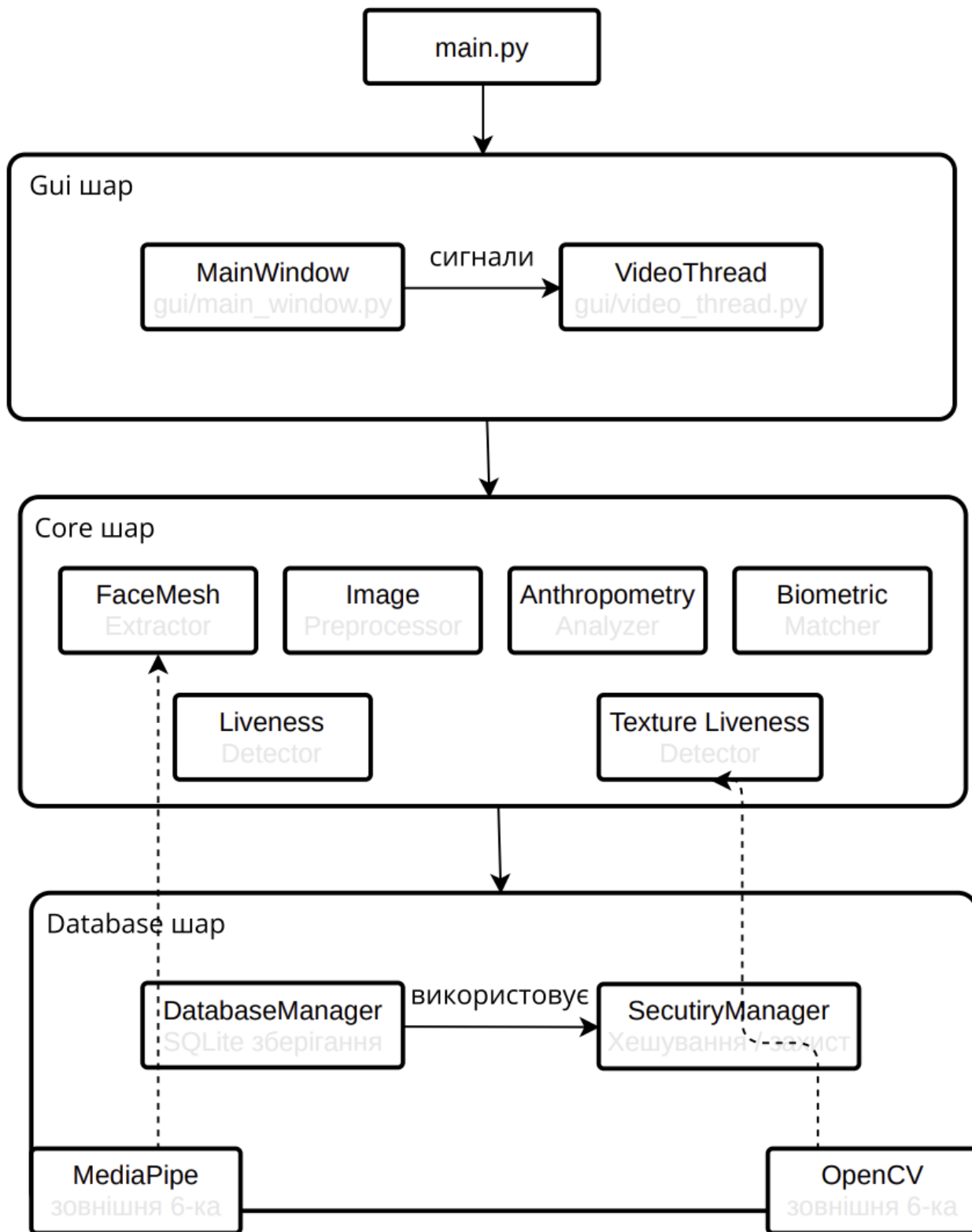
ДОДАТОК 1

**Інтелектуальна система біометричної автентифікації на основі
антропометрики обличчя**

**Схема структури системи
ІАЛЦ.467200.004 Д1**

Аркушів 1

Київ 2026 р



					ІАЛЦ.467200.004 Д1							
		№ докум.	Підпис	Дата								
Розробив		Крицька С. Д.			Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя Схема структури системи			Літ.	Аркуш	Аркушів		
Перевірив		Селіванов В. Л.								1	1	
Реценз.								КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-21				
Н. Контр.		Коренко Д. В.										
Затвердив		Волокита А. М.										

ДОДАТОК 2

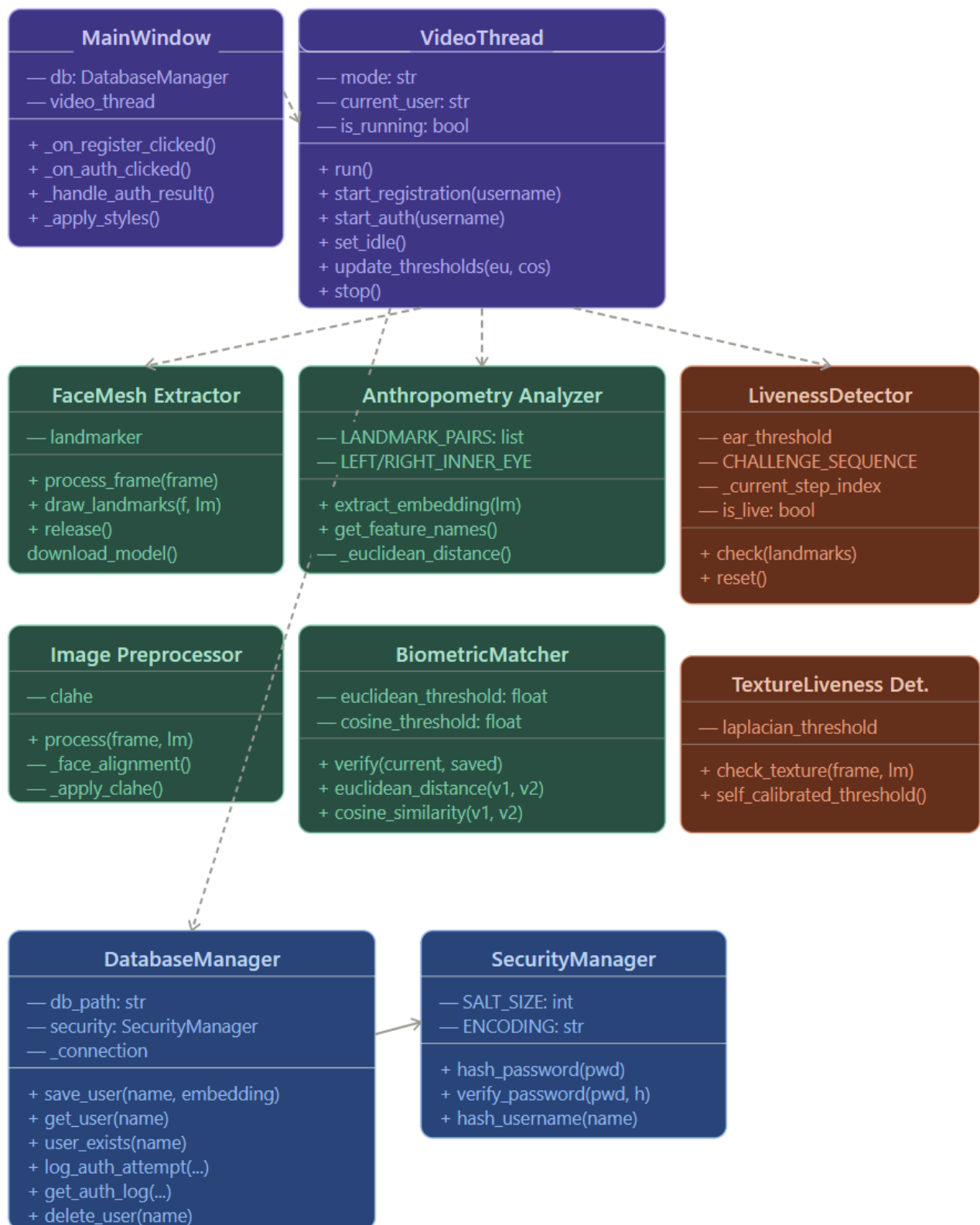
Інтелектуальна система біометричної автентифікації на основі
антропометрики обличчя

Функціональна схема (діаграма класів)

ІАЛЦ.467200.005 Д2

Аркушів 1

Київ 2026 р



					ІАЛЦ.467200.005 Д2							
		№ докум.	Підпис	Дата								
Розробив		Крицька С. Д.			Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя Функціональна схема (діаграма класів)			Літ.	Аркуш	Аркушів		
Перевірів		Селіванов В. Л.								1	1	
Реценз.								КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-21				
Н. Контр.		Коренко Д. В.										
Затвердив		Волокита А. М.										

ДОДАТОК 3

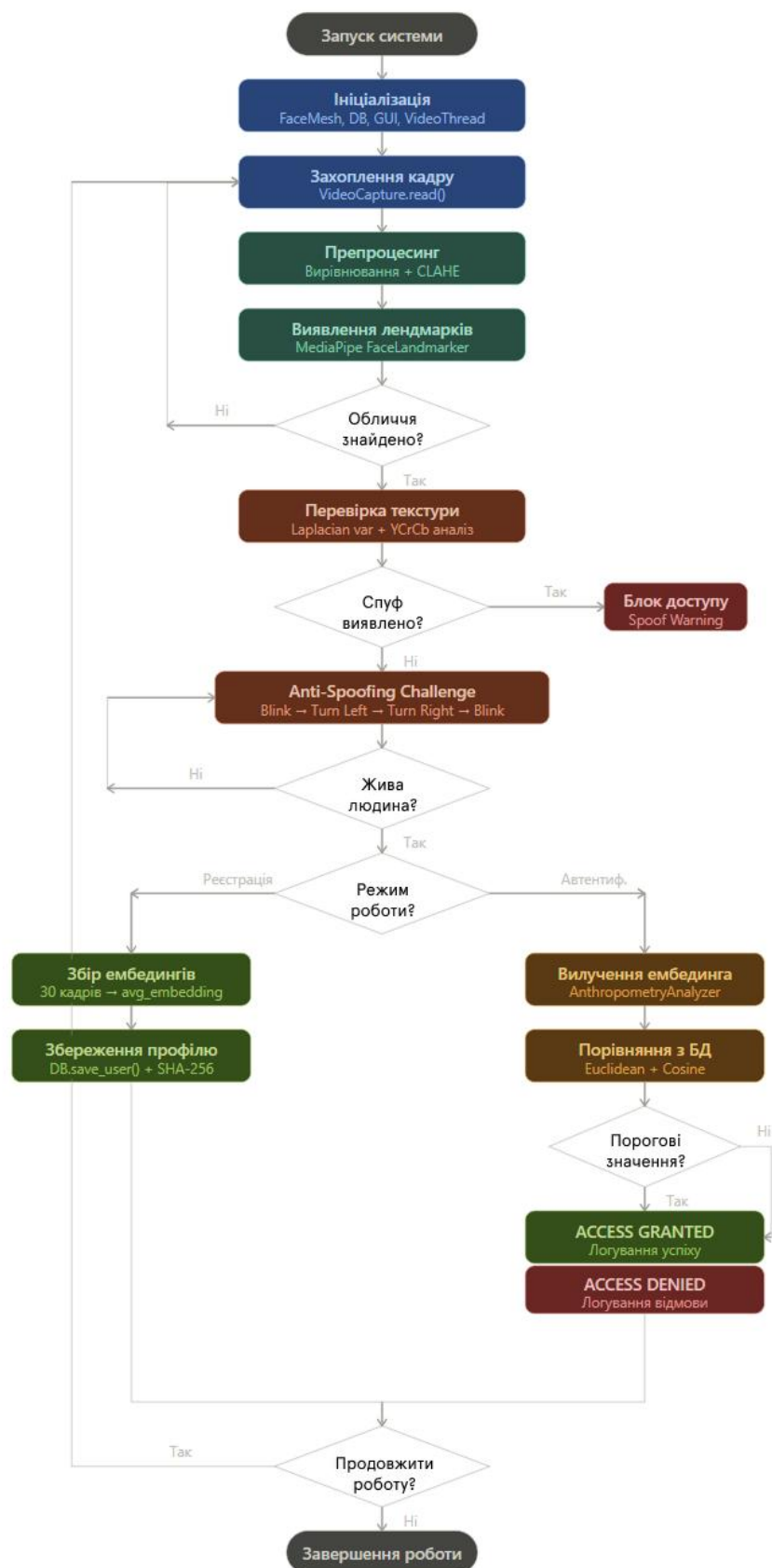
**Інтелектуальна система біометричної автентифікації на основі
антропометрики обличчя**

Послідовність дій програмного забезпечення

ІАЛЦ.467200.006 ДЗ

Аркушів 1

Київ 2026 р



ІАЛЦ.467200.006 ДЗ

					ІАЛЦ.467200.006 ДЗ				
		№ докум.	Підпис	Дата					
Розробив	Крицька С. Д.				Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя Послідовність дій програмного забезпечення	Літ.	Аркуш	Аркушів	
Перевірив	Селіванов В. Л.						1	1	
Реценз.						КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-21			
Н. Контр.	Коренко Д. В.								
Затвердив	Волокита А. М.								

ДОДАТОК 4

Інтелектуальна система біометричної автентифікації на основі
антропометрики обличчя

Текст програмного коду

ІАЛЦ.467200.007 Д4

Аркушів 1

Київ 2026

Репозиторій проєкту доступний на платформі GitHub [26]. Крім того, доступ забезпечено за QR кодом.



					ІАЛЦ.467200.007 Д4				
		№ докум.	Підпис	Дата					
Розробив	Крицька С. Д.				Інтелектуальна система біометричної автентифікації на основі антропометрики обличчя Текст програмного коду	Літ.	Аркуш	Аркушів	
Перевірив	Селіванов В. Л.						1	1	
Реценз.						КПІ ім. Ігоря Сікорського, ФІОТ, ІМ-			
Н. Контр.	Коренко Д. В.								
Затвердив	Волокита А. М.								