Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ СИСТЕМИ ІДЕНТИФІКАЦІЇ ОСОБИ З ВИКОРИСТАННЯМ БІОМЕТРИЧНОЇ ВЕРИФІКАЦІЇ ОБЛИЧЧЯ ТА ПЕРЕВІРКИ ДОКУМЕНТА В РЕЖИМІ РЕАЛЬНОГО ЧАСУ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-22-1
Матвєєв Є. Е.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Кобилін О. А.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Матвєєву Євгену Едуардовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення системи ідентифікації особи з використанням біометричної верифікації обличчя та перевірки документа в режимі реального часу

затверджена наказом університету від 26 травня 2026 року № 435Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 20 червня 2026 р.
3. Вихідні дані до роботи технічне завдання на розробку системи ідентифікації особи, методи та алгоритми комп'ютерного зору (OCR, біометрична верифікація), інструментальні засоби (Java 21, Python, JavaScript, Spring Boot 3, FastAPI), технології контейнеризації (Docker, Docker Compose), міжнародні стандарти (ICAO 9303, ISO/IEC 30107-3).

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз предметної області та систем віддаленої верифікації, обґрунтування вибору методів OCR та ідентифікації облич.
2 Моделювання алгоритмів перевірки живої присутності (Liveness Detection).
3. Проєктування архітектури розподіленої системи та бази даних.
4. Розроблення керівного (Java) і комп'ютерного зору (Python) сервісів.
5. Розроблення клієнтського застосунку та налаштування Webhooks.
6 Тестування системи.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) загальна схема процесу e-KYC, архітектурна діаграма взаємодії компонентів, блок-схеми алгоритмів (локалізація тексту, верифікація обличчя, детектор очей/Liveness), діаграми послідовності обміну даними, діаграми діяльності нейромережевого аналізу, діаграма розгортання інфраструктури, схеми клієнтського інтерфейсу, графіки та діаграми результатів тестування.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	13.04.2026	
2	Аналіз завдання, підбір літератури	14.04.26-16.04.26	
3	Аналіз літератури з досліджуваної проблеми	17.04.26-19.04.26	
4	Аналіз технічних засобів	20.04.26-22.04.26	
5	Розробка методу	23.04.26-05.05.26	
6	Програмна реалізація	26.04.26-20.05.26	
7	Оформлення пояснювальної записки	21.05.26-04.06.26	
8	Перевірка на нормоконтроль	05.06.26-09.06.26	
9	Перевірка на плагіат	10.06.26-19.06.26	
10	Рецензування	11.06.26-30.06.26	
11	Підготовка презентації та доповіді	20.06.26-30.06.26	
12	Занесення роботи в електронний архів	20.06.26-30.06.26	
	Попередній захист кваліфікаційної роботи	30.06.26-09.07.26	

Дата видачі завдання 13 квітня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____доц. Кобилін О. А._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 86 с., 14 табл., 28 рис., 50 джерел.

ВЕРИФІКАЦІЯ, МІКРОСЕРВІСНА АРХІТЕКТУРА, КОМП'ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ОБЛИЧ, ОПТИЧНЕ РОЗПІЗНАВАННЯ СИМВОЛІВ, JAVA, PYTHON.

Об'єктом роботи є процес віддаленої верифікації користувачів.

Метою роботи є розроблення програмного комплексу для автоматизованої верифікації особи (KYC) на базі мікросервісної архітектури та комп'ютерного зору.

Під час роботи використано: методи оптичного розпізнавання символів (OCR), аналіз біометричних даних, принципи проєктування розподілених систем та REST API.

Практична цінність одержаних результатів полягає у створенні масштабованого рішення для безпечної інтеграції із зовнішніми платформами без залучення комерційних аналогів.

Розроблено розподілену систему, що включає керуючий сервіс, спеціалізований модуль комп'ютерного зору та клієнтський вебзастосунок.

Область застосування: фінансові установи, сервіси електронної комерції, системи контролю доступу.

VERIFICATION, MICROSERVICE ARCHITECTURE, COMPUTER VISION, FACE RECOGNITION, OPTICAL CHARACTER RECOGNITION, JAVA, PYTHON.

The object of the work is the remote user verification process.

The goal of the work is to develop a software complex for automated user verification (KYC) based on microservice architecture and computer vision.

The following were used during the work: optical character recognition (OCR) methods, biometric data analysis, distributed systems design principles, and REST API.

The practical value of the obtained results lies in creating a scalable solution for secure integration with external platforms without involving commercial analogues.

A distributed system including a control service, a specialized computer vision module, and a client web application has been developed.

Applications: financial institutions, e-commerce services, access control systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AML – Anti-Money Laundering (протидія відмиванню грошей).

API – Application Programming Interface (прикладний програмний інтерфейс).

B2B – Business-to-Business (модель економічної взаємодії «бізнес для бізнесу»).

CRNN – Convolutional Recurrent Neural Network (згорткова рекурентна нейронна мережа).

e-KYC – Electronic Know Your Customer (електронна процедура віддаленої ідентифікації клієнта).

FATF – Financial Action Task Force (Міжнародна група з протидії відмиванню брудних грошей).

GDPR – General Data Protection Regulation (Загальний регламент Європейського Союзу про захист персональних даних).

ICAO – International Civil Aviation Organization (Міжнародна організація цивільної авіації).

KYC – Know Your Customer (нормативний принцип ідентифікації «Знай свого клієнта»).

MRZ – Machine Readable Zone (машинно-зчитувана зона документа, що посвідчує особу).

OCR – Optical Character Recognition (оптичне розпізнавання символів).

PAD – Presentation Attack Detection (виявлення атак на біометричне пред’явлення).

PEP – Politically Exposed Person (політично значуща особа).

SaaS – Software as a Service (програмне забезпечення як послуга).

SDK – Software Development Kit (набір засобів розроблення програмного забезпечення).

ВСТУП

Зростання рівня цифрового шахрайства та необхідність суворого дотримання нормативних вимог у фінансовому секторі (AML, KYC) зумовлюють надзвичайну актуальність завдань автоматизації перевірки особи. Процедура віддаленої верифікації користувачів стала обов'язковим стандартом для забезпечення безпеки доступу до сучасних інформаційних платформ. Традиційні методи ручної перевірки документів є ресурсомісткими та схильними до помилок у результаті впливу людського фактора. Відповідно до цього, впровадження новітніх технологій комп'ютерного зору дозволяє суттєво підвищити надійність та швидкість процесів ідентифікації.

Актуальність роботи полягає у необхідності створення доступних та масштабованих засобів для віддаленої ідентифікації користувачів в умовах постійного зростання цифрових загроз. Розроблення незалежної розподіленої системи дозволяє подолати обмеження, пов'язані з високою вартістю та закритою архітектурою комерційних аналогів. Таким чином, впровадження власного програмного комплексу забезпечує надійний захист персональних даних та гнучку інтеграцію для організацій різного масштабу.

Існуючі комерційні рішення часто виявляються економічно недоцільними для представників малого та середнього бізнесу через високу вартість транзакцій. Крім того, монополія корпоративних платформ із закритою архітектурою створює значні бар'єри під час інтеграції інформаційних систем. Вимоги щодо суворої локалізації персональних даних роблять використання сторонніх хмарних провайдерів юридично ризикованим кроком. Зазначені фактори формують технологічну прогалину та гостру потребу в розробленні відкритих, незалежних програмних рішень.

Практичне значення одержаних результатів полягає у створенні власного програмного комплексу, який поєднує можливості моделей штучного інтелекту з надійністю розподіленої архітектури. Розроблене

рішення усуває залежність від дорогих сторонніх сервісів і забезпечує повний контроль над процесом оброблення конфіденційної інформації. Запропонований комплекс пропонує гнучкість масштабування та простий інтерфейс для подальшої інтеграції у бізнес-процеси замовників. Створена система повністю готова до впровадження в інфраструктуру фінансових установ, сервісів оренди та електронної комерції.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз процесу віддаленої верифікації користувачів (KYC)

1.1.1 Еволюція процедури «Знай свого клієнта»

Стрімкий розвиток цифрової економіки та перехід комерційних послуг в Інтернет зумовили потребу в надійних механізмах ідентифікації особи. Процедура «Знай свого клієнта» (Know Your Customer, KYC) є обов'язковим стандартом для фінансових та комерційних установ. Її метою є запобігання шахрайству, відмиванню грошей (AML) та фінансуванню тероризму шляхом верифікації користувачів. Методи адаптивної кластеризації дозволяють ефективно виявляти аномалії в даних, що є критичним для систем KYC [1, 2]. Міжнародні стандарти FATF визначають основу для запобігання фінансовим злочинам.

Раніше верифікація вимагала фізичної присутності клієнта у відділенні для надання оригіналів документів, після чого співробітник візуально перевіряв особу та захисні елементи документів. Хоча такий підхід забезпечував високу безпеку, він став неефективним через цифровізацію послуг, географічні обмеження, високі витрати на утримання мережі відділень та тривалий час очікування, що змусило бізнес шукати нові шляхи [3].

Каталізатором переходу до віддаленої верифікації стали рекомендації FATF та європейські директиви AMLD4–AMLD6, що узаконили електронні засоби ідентифікації. Пандемія COVID-19 прискорила цей процес, унеможлививши традиційне обслуговування [4].

Перехід на e-KYC дозволив автоматизувати процес, мінімізувати витрати та забезпечити доступність послуг 24/7. Проте це створило нові виклики для кібербезпеки, захисту даних та точності алгоритмів. Зловмисники застосовують складні методи фальсифікації, такі як цифрова обробка документів, Deepfake-генерація осіб, а також використання

роздрукованих фото чи силіконових масок [5]. Загальну схему класичного процесу віддаленої верифікації показано на рисунку 1.1.

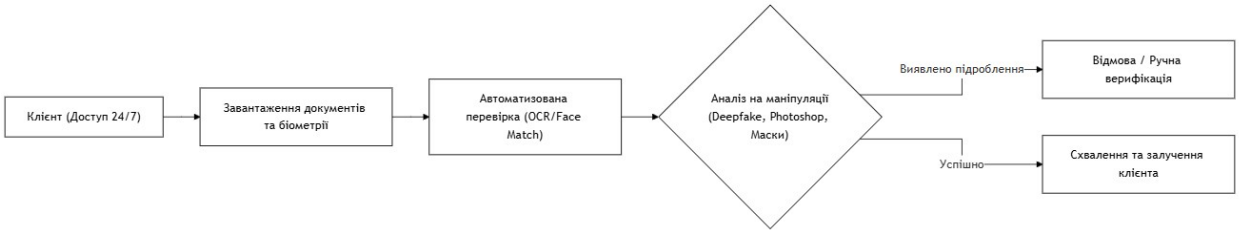

Рисунок 1.1 – Загальна схема класичного процесу віддаленої верифікації (e-KYC)

Сьогодні зовнішні інформаційні системи часто не мають ресурсів для створення складних алгоритмів верифікації документів із нуля. Розроблення власних модулів біометричного порівняння та оптичного розпізнавання потребує залучення вузькопрофільних фахівців, великих обсягів розмічених даних та значних обчислювальних потужностей. Через це виникає гостра потреба в інтеграції з готовими сервісами верифікації (IDVaaS). Основою для таких сервісів є використання надійних моделей ідентифікації нелінійних систем [6].

1.1.2 Огляд сучасних комерційних систем верифікації

Аналіз IT-ринку показує наявність потужних глобальних B2B-платформ віддаленої ідентифікації, серед лідерів яких є Onfido, SumSub, Veriff, Jumio та IDnow. Ці системи пропонують комплексний підхід до перевірки особи, використовуючи ансамблі моделей машинного навчання, бази втрачених документів та реєстри політично значущих осіб. Ефективність таких платформ значною мірою залежить від якості моделей класифікації та методів зниження обчислювальних витрат [7, 8].

Платформа SumSub забезпечує універсальне рішення для відповідності нормативним вимогам, підтримуючи документи понад 200 країн у більше ніж 6 000 форматів. Система містить модулі оптичного розпізнавання, перевірки біометрії та автоматичного пошуку за глобальними санкційними списками. При цьому використання методів стиснення описів дозволяє підвищити ефективність пошуку без втрати точності ідентифікації [9].

Компанія Onfido робить основний акцент на використанні штучного інтелекту для виявлення найскладніших спроб шахрайства. Алгоритми Onfido детально аналізують текстуру документа, перевіряють шрифти на відповідність державним стандартам та мають глибоко налаштовані механізми перевірки на життєздатність обличчя користувача під час запису короткого відео [10].

Система Veriff, у свою чергу, виділяється швидкою інтеграцією готового клієнтського застосунку. Вони пропонують автоматизовану відеоідентифікацію, яка фіксує не лише статичні кадри, але й поведінку користувача, інформацію про його пристрій та мережеве середовище, створюючи комплексний профіль ризику [11].

Спільними перевагами всіх розглянутих рішень є висока загальна швидкість оброблення типових запитів, наявність готових пакетів інструментів (SDK) для легкої інтеграції у мобільні застосункию.

Також вони повністю відповідають суворим міжнародним стандартам безпеки інформації (наприклад, ISO/IEC 27001, SOC 2 Type II). Проте, глибокий аналіз умов їх використання виявляє низку обмежень, що робить їх неприйнятними для багатьох проєктів.

Незважаючи на очевидні технологічні переваги, існуючі комерційні платформи мають суттєві недоліки, які створюють бар’єри для їх впровадження певними категоріями бізнесу. Насамперед, ключовою проблемою є висока вартість транзакцій. Більшість провайдерів використовують модель тарифікації оплати за фактичне споживання (Pay-As-You-Go). Така цінова політика робить використання цих систем

економічно недоцільним для невеликих локальних проєктів, освітніх платформ або стартапів.

Другою критичною проблемою є питання конфіденційності. Використання комерційних рішень у моделі «чорного ящика» передбачає передачу чутливих персональних даних користувачів на віддалені сервери третіх компаній. Це часто суперечить суворим внутрішнім політикам безпеки. Третім аспектом є надмірна складність інтеграції через необхідність встановлення громіздких монолітних пакетів інструментів розробника (SDK). Досвід використання складних нейромережевих архітектур, таких як моделі-трансформери, підтверджує необхідність ретельної оцінки ресурсних витрат під час інтеграції [12].

1.2 Огляд існуючих інформаційних технологій

Для успішного вирішення завдання автоматизованої верифікації особи критично важливим етапом є огляд та вибір оптимальних методів комп’ютерного зору. Аналіз сучасної наукової літератури, технічної документації та досліджень у галузі штучного інтелекту свідчить про те, що процес перевірки ідентичності фундаментально розбивається на два незалежних, але взаємодоповнюючих етапи: вилучення та аналіз текстової інформації з документа, що посвідчує особу, та підтвердження біометричної відповідності обличчя користувача з урахуванням перевірки на життєздатність [13].

1.2.1 Аналіз методів оптичного розпізнавання символів

Перший ключовий етап верифікації реалізується за допомогою технологій оптичного розпізнавання символів (Optical Character Recognition,

OCR) [14]. Історично розвиток цих систем пройшов шлях від простих алгоритмів зіставлення із шаблонами (Template Matching) до складних архітектур на базі глибинного навчання (Deep Learning).

У класичних системах процес розпізнавання тексту суворо поділявся на декілька послідовних кроків. Спочатку виконувалося попереднє оброблення зображення: переведення в градації сірого, адаптивна бінаризація для відокремлення тексту від фону, усунення цифрових шумів (медіанна фільтрація) та корекція кута нахилу документа (Deskewing). Далі відбувалася сегментація зображення на окремі текстові блоки, рядки, слова та в кінцевому підсумку на ізольовані символи [15]. Тільки після цього алгоритм намагався класифікувати кожен символ окремо. Найвідомішим представником класичного підходу є бібліотека Tesseract, початково розроблена компанією Hewlett-Packard, а згодом підтримана Google. Tesseract демонструє надзвичайно високі показники точності на ідеально відсканованих документах з контрастним чорним текстом на білому тлі. Проте аналіз публікацій останніх років доводить, що для фотографій документів, зроблених некваліфікованими користувачами на камери мобільних пристроїв при поганому освітленні, класичні методи працюють вкрай незадовільно. Наявність бліків від голограм, тіней, перспективних спотворень та складних фонових візерунків на сучасних паспортах призводить до масових помилок на етапі бінаризації та сегментації.

З огляду на ці проблеми, сучасна індустрія комп’ютерного зору повністю перейшла до використання нейромережевих архітектур для задачі розпізнавання тексту на природних зображеннях (Scene Text Recognition, STR). Сучасний конвеєр складається з двох нейронних мереж: моделі виявлення тексту (Text Detection) та моделі розпізнавання тексту (Text Recognition). Для виявлення текстових областей широко застосовуються архітектури EAST або CRAFT, які здатні знаходити текст будь-якої орієнтації та масштабу, генеруючи полігони обмежувальних рамок. Для етапу розпізнавання стандартом де-факто стали архітектури CRNN, що

поєднують згорткові мережі для вилучення ознак із рекурентними мережами для моделювання послідовностей [16].

У таблиці 1.1 наведено детальне порівняння найпопулярніших відкритих бібліотек для OCR, які придатні для використання у системах ідентифікації.

Для суттєвої оптимізації процесу та зниження відсотка помилок у системах e-KYC аналіз тексту зазвичай обмежується специфічними машинно-зчитуваними зонами (Machine Readable Zone, MRZ). Формат MRZ суворо регламентований стандартом Міжнародної організації цивільної авіації (ICAO) 9303.

Хоча ця організація безпосередньо опікується авіаційними питаннями, саме вона історично виступає глобальним регулятором стандартів для всіх машинозчитуваних проїзних та ідентифікаційних документів.

Метою цієї організації є забезпечення швидкого та універсального розпізнавання таких документів на прикордонних контролях через MRZ, що розташовується в нижній частині та містить найважливішу інформацію у закодованому вигляді. Для паспортів (формат TD3) це два рядки по 44 символи, для ID-карток (формат TD1) – три рядки по 30 символів. Перевага використання MRZ полягає у тому, що стандарт допускає використання лише обмеженого набору символів (латинські літери від A до Z, цифри від 0 до 9 та символ-заповнювач «<»).

Крім того, стандартом передбачено використання контрольних цифр (Check Digits), які обчислюються за спеціальним алгоритмом зваженої суми. Це дозволяє програмному комплексу не лише розпізнати текст, але й миттєво верифікувати його цілісність, виявляючи помилки алгоритму OCR або спроби грубої підробки документа (рис. 1.2). Застосування методів глибокого навчання до розпізнавання об’єктів різної природи демонструє високу універсальність таких конвеєрів [17].

Таблиця 1.1 – Порівняння відкритих бібліотек оптичного розпізнавання тексту

Назва бібліотеки	**Базова архітектура / Технологія**	**Основні переваги підходу**	**Виявлені недоліки та обмеження**
Tesseract OCR	LSTM (Довга коротка пам’ять), класичний конвеєр	Мінімальні вимоги до ресурсів (працює на CPU), стабільність, відкритий вихідний код	Критична вразливість до тіней, бліків та зміни перспективи на фотографіях з мобільного
EasyOCR	PyTorch (використовує CRAFT для виявлення та CRNN для розпізнавання)	Висока точність на фотографіях зі складним фоном, підтримка понад 80 мов з коробки	Високе споживання оперативної пам’яті, низька швидкість без використання графічних процесорів (GPU)
PaddleOCR	Фреймворк PaddlePaddle, ансамбль оптимізованих моделей глибинного навчання	Баланс між високою швидкістю та точністю, наявність моделей розміром менше 10 МБ	Складність початкового налаштування середовища, менша кількість документації

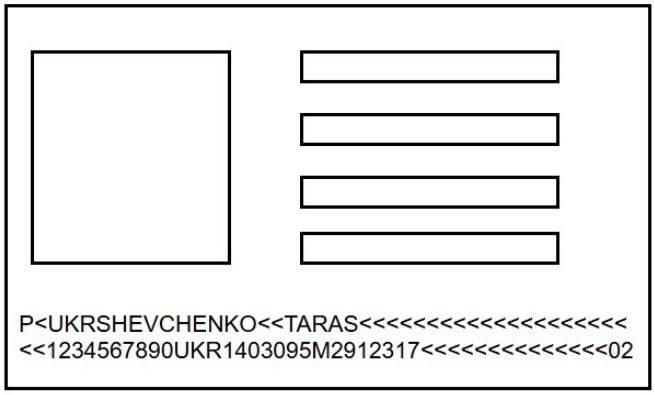

Рисунок 1.2 – Приклад машинно-зчитуваної зони (MRZ) документа, що посвідчує особу

1.2.2 Основи порівняння біометричних векторів

Другим етапом верифікації є біометричний аналіз обличчя. Наукова проблема біометричної ідентифікації досліджується десятиліттями, еволюціонуючи від математичних методів головних компонент (PCA), відомих як власні обличчя (Eigenfaces), до сучасних глибоких згорткових нейронних мереж. Глобально завдання поділяється на кілька підзадач: виявлення обличчя на вхідному зображенні (Face Detection), вирівнювання обличчя за ключовими точками (Face Alignment) та безпосередньо порівняння (Face Verification).

На етапі виявлення система має локалізувати обличчя незалежно від його розміру, кута нахилу та наявності перешкод (наприклад, медичних масок чи окулярів). У той час як раніше домінував метод каскадів Хаара (Haar Cascades), сьогодні беззаперечними лідерами є нейромережеві детектори, такі як MTCNN, RetinaFace або YOLO-Face. Вони не лише виділяють обмежувальну рамку, але й визначають координати антропометричних точок (центрів очей, кінчика носа, куточків рота). Використовуючи ці точки, алгоритм здійснює афінне перетворення, щоб очі завжди знаходилися на одній горизонтальній лінії [18, 19]. Це стандартизує

вхідні дані для наступного кроку. Подібні методи аналізу продуктивності алгоритмів класифікації знаходять широке застосування у сучасних системах комп’ютерного зору [20].

Етап порівняння ґрунтується на концепції перетворення зображення обличчя у вектор ознак (Face Embedding) фіксованої розмірності. Сучасні архітектури, такі як FaceNet, SphereFace або ArcFace, навчаються на гігантських базах даних облич з метою формування такого математичного простору, де вектори фотографій однієї людини знаходяться максимально близько один до одного, а вектори різних людей – максимально далеко.

Процес порівняння у розподіленій системі зводиться до обчислення математичної міри подібності між вектором, отриманим з фотографії у паспорті, та вектором, отриманим із селфі користувача. У науковій літературі найчастіше розглядаються дві метрики: евклідова відстань та косинусна подібність. Евклідова відстань дозволяє оцінити абсолютну лінійну різницю між векторами у багатовимірному просторі. Натомість косинусна подібність оцінює кут між векторами, ігноруючи їхню абсолютну довжину. Згідно з дослідженнями у галузі комп’ютерного зору, саме косинусна подібність є більш стійким і надійним показником при зміні умов освітленості обличчя на різних фотографіях [21].

Якщо обчислене значення близькості (подібності) перевищує заздалегідь встановлене порогове значення (Threshold), система приймає автоматичне рішення про те, що на обох фотографіях зображена одна й та сама людина. Детальне математичне обґрунтування цих метрик, а також специфіки функцій втрат для навчання таких моделей, є предметом подальшого дослідження на етапі проєктування системи. Аналогічні підходи до розпізнавання емоцій за допомогою згорткових мереж також ґрунтуються на аналізі динамічних мімічних особливостей [22].

1.2.3 Алгоритми перевірки життєздатності

Наявність точного алгоритму порівняння облич не гарантує безпеки системи без впровадження механізмів виявлення атак на біометричне пред'явлення (Presentation Attack Detection, PAD), які у комерційній сфері називають перевіркою життєздатності (Liveness Detection). Без цієї перевірки зловмисник може обдурити систему, просто показавши перед камерою знайдену в інтернеті фотографію жертви [23].

Методологія та вимоги до систем PAD регламентуються міжнародним стандартом ISO/IEC 30107-3. Згідно зі стандартом, атаки поділяються на декілька рівнів складності [24]. До простих атак (Level 1) належать роздруковані на папері фотографії або показ зображення з екрана іншого смартфона. До складніших (Level 2 та Level 3) – відтворення заздалегідь записаного відео, використання тривимірних силіконових масок або застосування програмного забезпечення для заміни обличчя в реальному часі (Deepfake/Virtual Camera).

Методи протидії традиційно поділяються на активні та пасивні. Активні методи вимагають від користувача виконання певних, згенерованих випадковим чином інструкцій перед камерою: посміхнутися, кліпнути очима, повернути голову ліворуч чи праворуч, або вимовити певну фразу. Незважаючи на свою поширеність, активні методи мають суттєві недоліки. Вони погіршують користувацький досвід (UX), збільшують час проходження верифікації і, що найважливіше, стають вразливими до сучасних Deepfake-технологій, які здатні імітувати рухи голови та міміку в реальному часі [25].

Сучасні наукові розробки фокусуються на пасивних методах перевірки життєздатності. Пасивний метод (Passive Liveness) не вимагає від клієнта жодних додаткових дій. Система самостійно аналізує один або кілька статичних кадрів за допомогою спеціалізованих нейронних мереж. Аналіз ґрунтується на пошуку мікроскопічних артефактів: відблисків від скла екрана

або фотопаперу (Moire pattern), неприродної текстури шкіри, відсутності тривимірної глибини сцени або розмиття контурів, що виникає при спробі обрізати маску. Деякі передові підходи використовують аналіз відбиття світла від обличчя екраном пристрою (Flash Liveness). Враховуючи необхідність забезпечення швидкого та зручного процесу, використання пасивних методів є пріоритетним напрямком при проєктуванні сучасних систем верифікації [26].

1.3 Аналіз архітектурних рішень

Проєктування програмного комплексу, що працює з мультимедійними даними та алгоритмами машинного навчання у режимі реального часу, вимагає ретельного підбору архітектурних шаблонів. Згідно з сучасними практиками інженерії програмного забезпечення, вибір базової архітектури здійснюється між монолітним підходом та мікросервісною моделлю [27].

Монолітна архітектура передбачає жорстке об’єднання всіх функціональних компонентів системи у єдиний виконуваний процес. Проте для системи комп’ютерного зору цей підхід має критичні недоліки. Завдання роботи нейронних мереж є інтенсивними обчислювальними операціями. Якщо сервер одночасно прийматиме мережеві запити і намагатиметься виконувати ресурсомісткі математичні операції над матрицями пікселів, це неминуче призведе до вичерпання пулу потоків та зупинки системи [28]. Порівняльний аналіз архітектурних підходів наведено у таблиці 1.2.

Огляд науково-технічної літератури свідчить, що для розв’язання подібних завдань найбільш доцільно розглядати розподілену мікросервісну архітектуру. У класичних теоретичних моделях таких систем зазвичай виділяють основний вузол управління (наприклад, API Gateway) та набір ізольованих обчислювальних вузлів.

Таблиця 1.2 – Порівняльний аналіз архітектурних підходів для систем аналізу медіаданих

Критерій оцінювання	**Монолітна архітектура**	**Мікросервісна архітектура**
Масштабування обчислень	Вимагає дублювання всієї системи, неефективне використання ОЗП	Дозволяє точково збільшувати кількість екземплярів лише модулів AI
Технологічна гнучкість	Усі модулі пишуться однією мовою (компроміси у продуктивності)	Поліглотний підхід: вибір найкращої мови для конкретної задачі
Стійкість до збоїв	Помилка OCR зупиняє весь вебсервер	Падіння вузла обробки фото не зупиняє створення нових сесій

Згідно з проаналізованими джерелами, роль оркестратора переважно зводиться до керування життєвим циклом сесій і маршрутизації трафіку, тоді як ізольовані вузли беруть на себе ресурсомісткі завдання оброблення зображень та роботи нейронних мереж. Узагальнену теоретичну архітектурну діаграму взаємодії таких типових компонентів показано на рисунку 1.3.

Окрім компонентного розподілу, важливим аспектом інтеграції є спосіб інформування сторонніх платформ. Оскільки процес розпізнавання мультимедійних даних триває певний час, утримання відкритого синхронного HTTP-з’єднання є архітектурно нераціональним. Періодичне опитування стану (Long Polling) призводить до марнотратного використання мережевих ресурсів. Тому, як показує аналіз, архітектура повинна базуватися на подійно-орієнтованій, асинхронній комунікації за допомогою механізму

зворотних викликів (Webhook). Цей шаблон дозволяє зовнішній системі ініціювати перевірку, розірвати з'єднання та пасивно очікувати фінальних результатів від оркестратора [29, 30].

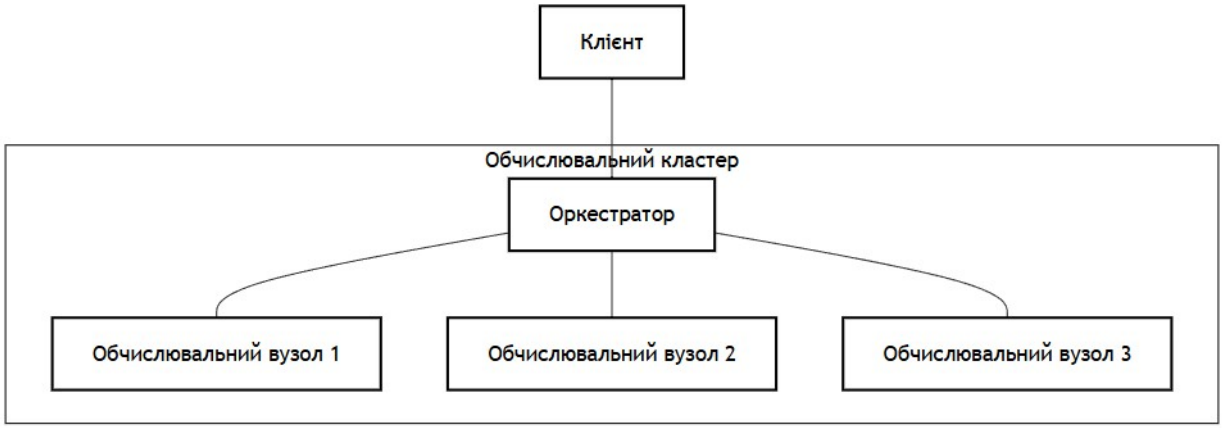

Рисунок 1.3 – Архітектурна діаграма взаємодії компонентів розподіленої системи верифікації

1.4 Постановка задачі

Зростання рівня цифрового шахрайства та необхідність дотримання нормативних вимог у фінансовому секторі (AML, KYC) роблять завдання автоматизації перевірки особи актуальним. Існуючі комерційні рішення часто є економічно недоцільними для малого та середнього бізнесу через високу вартість транзакцій. Крім того, монополія корпоративних платформ із закритою архітектурою створює бар'єри інтеграції, а вимоги щодо локалізації персональних даних роблять використання сторонніх хмарних провайдерів юридично ризикованим. Це зумовлює технологічну прогалину та потребу в пошуку відкритих, незалежних рішень.

Відповідно, розроблення власного програмного комплексу, який поєднує можливості моделей комп'ютерного зору з надійністю розподіленої мікросервісної архітектури, становить значний практичний і науковий

інтерес. Це дозволить створити систему, що забезпечує гнучкість масштабування, захист конфіденційної інформації та простий інтерфейс для інтеграції.

Об’єктом роботи є процес віддаленої верифікації користувачів.

Метою роботи є розроблення програмного комплексу для автоматизованої верифікації особи (KYC) на базі мікросервісної архітектури та комп’ютерного зору.

Для досягнення поставленої мети необхідно вирішити такі науково-практичні завдання:

− проаналізувати предметну область, дослідити процеси віддаленої ідентифікації та виявити проблеми використання комерційних систем;

− здійснити критичний аналіз алгоритмів комп’ютерного зору для задач вилучення тексту з фотографій документів (OCR) та біометричної верифікації облич користувачів;

− дослідити та порівняти існуючі архітектурні шаблони, обґрунтувати доцільність застосування розподіленої мікросервісної інфраструктури;

− спроєктувати загальну архітектуру програмної системи, визначивши ролі кожного компонента та протоколи їх взаємодії;

− розробити основний керуючий сервіс для управління сесіями та взаємодії із зовнішніми системами за допомогою механізму зворотних викликів (webhook);

− розробити ізольований сервіс комп’ютерного зору для оброблення мультимедійних даних;

− розробити клієнтський вебзастосунок для збору фотографій документа та обличчя за допомогою камери пристрою кінцевого користувача;

2 ОБГРУНТУВАННЯ МЕТОДІВ ТА ПРОЄКТУВАННЯ СИСТЕМИ

У цьому розділі проводиться детальний аналіз та наукове обґрунтування вибору методів комп’ютерного зору для розв’язання задач автоматизованої верифікації. Виконується математичне моделювання процесів оптичного розпізнавання символів та біометричної ідентифікації обличчя [31]. Розглядаються теоретичні основи алгоритмів перевірки живої присутності користувача в кадрі з використанням антропометричних метрик. На основі проведеного аналізу формуються функціональні та нефункціональні вимоги до компонентів системи. Здійснюється об’єктно-орієнтоване моделювання архітектурних рішень для реалізації розподіленого програмного комплексу.

2.1 Обґрунтування вибору методів комп’ютерного зору

Завдання автоматизованої верифікації особи вимагає послідовного розв’язання двох фундаментальних підзадач. Першою підзадачею є вилучення текстової інформації з візуальних зображень документів з використанням інформаційних технологій оптичного розпізнавання символів. Другою підзадачею є безпосередня біометрична ідентифікація обличчя шляхом математичного порівняння захопленого зображення користувача з еталонним фото на документі.

2.1.1 Вибір методу оптичного розпізнавання символів

Для забезпечення високої точності зчитування даних з документів, що посвідчують особу, здійснено порівняльний аналіз класичних методів на основі контурного аналізу та сучасних нейромережевих підходів. Усталені

рішення демонструють високу ефективність при роботі з ідеально відсканованими документами з високим рівнем контрастності. Проте у контексті систем віддаленої верифікації зображення зазвичай отримуються з камер мобільних пристроїв, що зумовлює наявність відблисків, геометричних викривлень, низької роздільної здатності та складного фону.

Класичні методи алгоритмічної бінаризації зображення в таких умовах призводять до втрати критично важливої інформації, що знижує точність розпізнавання машинозчитуваної зони (MRZ) до неприйнятного рівня [32]. З огляду на зазначені обмеження, обґрунтовано використання нейромережевої архітектури CRNN (Convolutional Recurrent Neural Network). Теоретичне обґрунтування вибору базується на здатності алгоритму поєднувати переваги двох фундаментальних типів штучних нейронних мереж. На першому етапі використовуються згорткові шари (CNN) для автоматичного вилучення просторових ознак із зображення. Математично операція двовимірної згортки для вхідного зображення I та ядра згортки K описується рівнянням:

$$S(i,j) = (I \cdot K)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n), \tag{2.1}$$

де $S(i,j)$ – результат згортки для конкретної позиції (i,j);

$I(i+m, j+n)$ – поточна ділянка вхідного зображення, яку зараз перекриває ядро;

$K(m,n)$ – вагові коефіцієнти самого фільтра.

Цей підхід дозволяє ідентифікувати локальні шаблони, зокрема грані, кути та структурні елементи специфічних символів шрифту OCR-B. На другому етапі послідовність вилучених просторових ознак передається до рекурентних шарів (LSTM), які здатні враховувати контекстуальну залежність між сусідніми символами. Оптимізація моделі відбувається шляхом мінімізації функції втрат CTC (Connectionist Temporal Classification),

яка дозволяє навчати мережу без необхідності точного посимвольного вирівнювання набору даних [33]. Функція втрат CTC визначається як від'ємний логарифм ймовірності цільової послідовності за умови вхідної послідовності згідно з рівнянням:

$$L_{CTC} = -\ln\left(P\left(Y\middle|X\right)\right) = -\ln\left(\sum_{\pi\in\beta^{-1}(Y)} P\left(\pi\middle|X\right)\right), \tag{2.2}$$

де Y – цільова послідовність токенів;

X – вхідна послідовність ознак;

$\beta^{-1}(Y)$ – множина всіх можливих шляхів π, що після видалення.

дублікатів та порожніх символів відповідають цільовій послідовності Y.

Для локалізації текстових зон перед їх безпосереднім розпізнаванням доцільно використовувати метод CRAFT [16]. Цей алгоритм генерує дві теплові карти: карту ймовірності знаходження центру символу та карту спорідненості між сусідніми символами. Зв'язок між етапами обробки документа наведено на відповідній блок-схемі (рис. 2.1).

Рисунок 2.1 – Загальна схема алгоритму локалізації та розпізнавання тексту

Такий підхід дозволяє системі визначати межі слів навіть за наявності значних перспективних викривлень, які виникають при нахилі камери. Використання архітектури CRNN забезпечує необхідний рівень надійності вилучення даних, нівелюючи негативний вплив факторів зовнішнього середовища під час процесу віддаленої перевірки [34].

2.1.2 Математичне обґрунтування методу ідентифікації облич

Процес підтвердження того, що особа є власником наданого документа, вимагає розв’язання задачі біометричного порівняння двох зображень. Складність цього процесу полягає в тому, що фотографія в документі є статичною та зробленою за оптимальних умов освітлення в минулому, тоді як поточне зображення піддається впливу динамічного освітлення та вікових змін. Використання класичного порівняння за ключовими точками в даному випадку є неефективним через неможливість точного зіставлення геометрії на пласкому скані документа [35].

Для розв’язання цієї проблеми обґрунтовано застосування методу глибинного метричного навчання на базі архітектури залишкової нейронної мережі ResNet-34. Сутність цього методу полягає у відображенні зображення в багатовимірний гіперпростір ознак, де нейронна мережа генерує унікальний вектор (ембединг). Мережа навчається таким чином, щоб вектори однієї особи групувалися максимально близько один до одного, а вектори різних осіб відповідно відштовхувалися. Процес навчання базується на мінімізації функції втрат триплету (Triplet Loss):

$$\ell = \max\left(0, d\left(A,P\right) - d\left(A,N\right) + \alpha\right), \tag{2.3}$$

де A – еталонне зображення;

P – інше зображення тієї ж особи;

N – зображення іншої особи;

d – функція обчислення дистанції;

α – гіперпараметр відступу.

На етапі практичної експлуатації системи обчислюється відстань між отриманими векторами U та D. Окрім Евклідової відстані, для оцінки подібності векторів у багатовимірному просторі доцільно застосовувати

метрику косинусної подібності, яка вимірює косинус кута між двома векторами і є інваріантною до їхньої абсолютної довжини:

$$S_c(U,D) = \frac{U \cdot D}{\|U\|\|D\|} = \frac{\sum_{i=1}^{n} u_i d_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}}. \quad (2.4)$$

Значення $S_c(U,D)$ наближається до 1, якщо вектори ідентичні, та до 0 у разі їхньої ортогональності. Для автоматизації ухвалення рішень встановлюються критичні пороги відстані, які впливають на показники хибного пропуску (FAR) та хибної відмови (FRR). Для систем безпеки мінімізація показника FAR є пріоритетною [36]. Загальна послідовність етапів ідентифікації обличчя відображена за допомогою відповідної діаграми алгоритму (рис. 2.2).

Рисунок 2.2 – Схема алгоритму біометричної верифікації обличчя

Багаторівнева порогова модель дозволяє системі працювати в автоматичному режимі для переважної більшості запитів [37]. Водночас забезпечується надійний механізм перенаправлення складних випадків для

ручного аналізу, що відповідає сучасним вимогам проєктування інформаційних систем.

2.2 Моделювання перевірки живої присутності

Системи віддаленої верифікації піддаються високому ризику атак із використанням засобів імітації (Presentation Attack). Для здійснення несанкціонованого доступу можуть використовуватися роздруковані статичні фотографії, екрани мобільних пристроїв або заздалегідь записані відеоматеріали. Для протидії цим загрозам розроблено алгоритм визначення живої присутності, що базується на аналізі динамічних фізіологічних характеристик та геометричних властивостей об’єкта перед камерою. Метод не потребує використання спеціалізованого апаратного забезпечення, спираючись виключно на математичний аналіз відеопотоку [38].

2.2.1 Аналіз антропометричних пропорцій

Базовим методом перевірки активної живої присутності є аналіз мікрорухів обличчя, зокрема детектування процесу кліпання очима. Здійснення цієї дії є рефлекторним, і її відтворення за допомогою статичних зображень неможливе. Для математичного моделювання цього процесу застосовується алгоритм відстеження шістдесяти восьми ключових точок обличчя [20]. Система ідентифікує координати контурів очей на кожному кадрі. Для обчислення ступеня розкриття ока використовується коефіцієнт співвідношення сторін ока (EAR), що базується на відношенні відстаней між вертикальними та горизонтальними ключовими точками повік $p_2,...,p_6$:

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}. \quad (2.5)$$

Динаміка зміни коефіцієнта у часі є надійним індикатором кліпання. Коли око відкрите, значення залишається стабільним у межах від 0,25 до 0,35. Під час кліпання значення різко падає і наближається до нуля. Аналізується часовий ряд значень $EAR(t)$. Для виключення випадкових шумів запроваджено математичну умову тривалості кліпання. Акт кліпання фіксується лише за умови, що час перебування значення EAR нижче критичного порогу $\triangle t_{blink}$ задовольняє умову природного рефлексу [39]:

$$T_{\min} \le (t_{end} - t_{start}) \le T_{\max}, \quad (2.6)$$

де $T_{\min}$ – мінімальна тривалість природного кліпання (зазвичай від 100 до 400 мілісекунд);

$T_{\max}$ – максимальна тривалість природного кліпання.

Логіку прийняття рішень щодо визначення кліпання змодельовано у вигляді алгоритмічного графа (рис. 2.3).

Рисунок 2.3 – Схема переходів станів детектора очей на основі метрики EAR

Використання нормалізованого відношення робить цей метод інваріантним до відстані користувача до об’єктива та кута нахилу голови. Застосування часових фільтрів дозволяє ефективно відрізняти природне кліпання від артефактів стиснення відеопотоку.

2.2.2 Геометричний аналіз повороту голови

Метод аналізу EAR є вразливим до атак із використанням вирізаних масок або планшетів із відеозаписами. Додатковим етапом перевірки є пасивний геометричний аналіз, що базується на ефекті паралакса. Метод дозволяє довести, що об’єкт є тривимірною структурою, а не пласкою проєкцією. Методи аналізу відеопотоків для підрахунку об’єктів у реальному часі підтверджують ефективність просторового розділення планів [40]. Об’ємне обличчя має виражений рельєф, де центральна точка (кінчик носа) знаходиться ближче до об’єктива, ніж краї щелепи. При мікрорухах голови проєкції цих точок зміщуються на матриці з різною швидкістю. Натомість при зміні кута нахилу смартфона зі статичним фото всі точки зміщуються синхронно.

Для математичної оцінки обчислюється динамічне співвідношення відстаней від центральної точки до периферійних. Відношення відстані до лівого ока D_{left} та правого ока D_{right} позначається як $R = D_{left} / D_{right}$. При зміні ракурсу реальної голови проєкція відстані до одного з очей зменшується значно швидше, що призводить до суттєвої зміни коефіцієнта R. Для формалізації критерію об’ємності система розраховує дисперсію коефіцієнта R на інтервалі часу $\triangle t$, що складається з N кадрів:

$$\sigma_R^2 = \frac{1}{N}\sum_{i=1}^{N}\left(R_i - \bar{R}\right)^2 . \qquad (2.7)$$

Якщо дисперсія σ_R^2 наближається до нуля під час детектування загального руху об’єкта в кадрі, робиться висновок про відсутність ефекту паралакса. Для підвищення точності запроваджено розрахунок нормалізованої варіації відстані:

$$\Delta R = \frac{\left| R_t - R_{t-1} \right|}{R_{t-1}}. \tag{2.8}$$

Перевищення показником ΔR встановленого порогу при повороті голови підтверджує тривимірність об’єкта. Комбінація методів оцінки EAR та аналізу паралакса забезпечує високу надійність підсистеми без потреби в інфрачервоних сенсорах. Застосування систем ортогональних функцій дозволяє сформувати компактний простір ознак для надійного розпізнавання [41].

2.3 Формування вимог до системи верифікації

На основі обраних математичних моделей та алгоритмів комп’ютерного зору сформовано комплекс вимог до програмного забезпечення. Процес інженерії вимог базується на стандарті якості програмного забезпечення ISO/IEC 25010. Для забезпечення модульності та інтегрованості розподіленої системи загальні вимоги були декомпоновані та деталізовані для кожного мікросервісу окремо [42].

Першочерговим обмеженням для всієї системи є загальний час обробки запиту. Сумарний час очікування для користувача T_{total} є сумою витрат часу на різних етапах обробки:

$$T_{total} = T_{net} + T_{arch} + T_{cv} \le T_{\max}, \tag{2.9}$$

де T_{net} – час мережевої затримки;

T_{arch} – час обробки на керуючому сервісі;

T_{cv} – час нейромережевих обчислень;

$T_{\max}$ – гранично допустимий час;

Для дотримання гранично допустимого часу $T_{\max}$ (встановленого на рівні 7 секунд) розроблено модель розподілу часового бюджету між компонентами. У перспективі продуктивність системи може бути додатково оптимізована шляхом комбінованого використання великих мовних моделей [43]. Зазначений розподіл обмежень цільового часу обробки (SLA) візуалізовано за допомогою діаграми послідовності на рисунку 2.4.

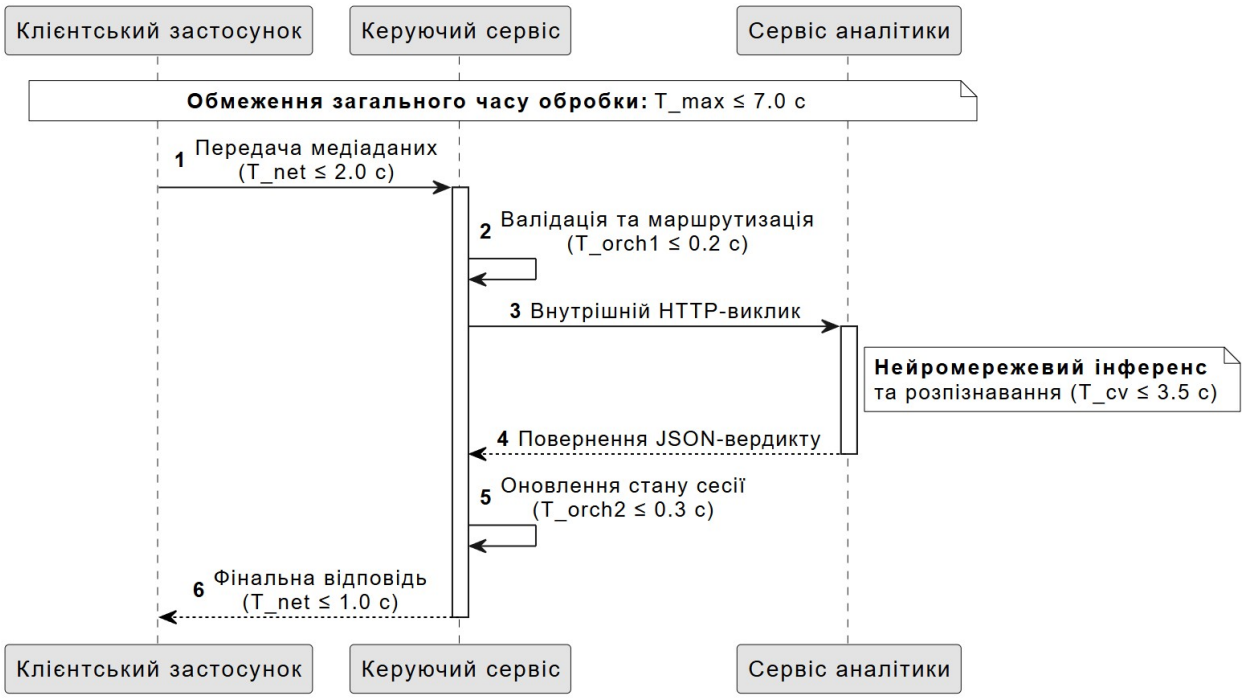

Рисунок 2.4 – Розподіл цільового часового бюджету між етапами взаємодії мікросервісів

Для досягнення вказаних показників систему розділено на три ізольовані компоненти, до кожного з яких висуваються специфічні функціональні та нефункціональні вимоги.

2.3.1 Вимоги до клієнтського застосунку

Клієнтський застосунок функціонує як основний інтерфейс збору первинних біометричних даних. Його архітектурна мета полягає у перенесенні частини базових обчислень на пристрій користувача для зменшення мережевого навантаження. Розподіл відповідальності компонента наведено на рисунку 2.5.

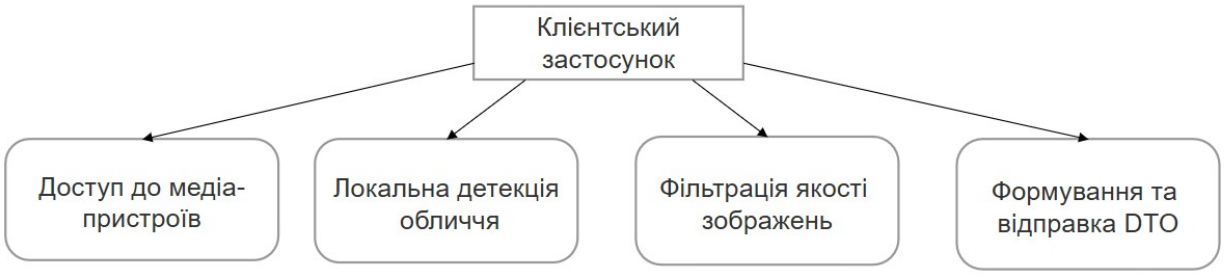

Рисунок 2.5 – Зони відповідальності клієнтського застосунку

Функціональні вимоги до цього компонента зосереджені навколо взаємодії з апаратним забезпеченням та підготовки даних до відправлення (табл. 2.1).

Таблиця 2.1 – Функціональні вимоги до клієнтського застосунку

ID	**Назва вимоги**	**Детальний опис функціональності**
1	2	3
ФВ-К-01	Доступ до вебкамери	Застосунок зобов’язаний отримувати відеопотік з камери пристрою через API WebRTC без використання сторонніх плагінів.
ФВ-К-02	Візуальна навігація	Інтерфейс повинен містити динамічні напрямні рамки (оверлеї) для орієнтування користувача під час позиціонування документа.

Продовження таблиці 2.1

1	2	3
ФВ-К-03	Локальне виявлення	Перед фіксацією кадру система має локально виявити наявність обличчя, запобігаючи відправленню порожніх зображень.
ФВ-К-04	Упакування DTO	Компонент має об’єднувати зображення та масиви координат у єдиний пакет формату multipart/form-data.

Нефункціональні вимоги до клієнтського застосунку регламентують його доступність, оптимізацію мережевого трафіку та кросплатформність (табл. 2.2).

Таблиця 2.2 – Нефункціональні вимоги до клієнтського застосунку

ID	**Категорія**	**Цільова метрика / Обмеження**	**Опис вимоги**
1	2	3	4
НФ-К-01	Мережева ефективність	≤ 5 Мегабайтів	Розмір сформованого пакета даних не повинен перевищувати ліміт завдяки попередньому стисненню.
НФ-К-02	Кросплатформність	HTML5 / WebRTC	Гарантована працездатність у стандартних веббраузерах на платформах iOS, Android, Windows та macOS.

Продовження таблиці 2.1

1	2	3	4
НФ-К-03	Ергономічність	≤100 мілісекунд	Час реакції інтерфейсу на дії користувача (натискання кнопок, зміна станів) має бути непомітним.
НФ-К-04	Безпека передачі	TLS 1.2+	Увесь трафік між клієнтом та сервером повинен передаватися виключно через зашифровані канали зв’язку.

2.3.2 Вимоги до керуючого сервісу

Керуючий сервіс виконує роль системного оркестратора та центрального шлюзу. До його обов’язків входить управління транзакціями, перевіряння прав доступу та забезпечення комунікації із зовнішніми платформами. Структуру функціональних завдань керуючого сервісу відображено на рисунку 2.6.

Рисунок 2.6 – Зони відповідальності керуючого сервісу

Функціональні вимоги до керуючого сервісу сфокусовані на координації процесів та забезпеченні життєвого циклу сесій (табл. 2.3). Серед нефункціональних вимог для цього компонента критичною є масштабованість. Згідно із законом Літла ($L = \lambda \cdot W$) з теорії масового обслуговування [24], для обробки $\lambda = 50$ запитів на секунду при середньому

часі життя сесії $W = 120$ секунд, сервіс повинен гарантувати стабільність роботи (табл. 2.4).

Таблиця 2.3 – Функціональні вимоги до керуючого сервісу

ID	**Назва вимоги**	**Детальний опис функціональності**
1	2	3
ФВ-О-01	Ініціалізація сесій	Прийом POST-запитів від зовнішніх систем, створення унікальних сесій та криптографічних маркерів.
ФВ-О-02	Валідація токенів	Перевірка автентичності токенів та їхнього терміну дії (TTL) перед наданням доступу клієнтському застосунку.
ФВ-О-03	Управління станами	Зміна стану сесії у процесі верифікації для блокування повторних спроб відправлення даних.
ФВ-О-04	Маршрутизація даних	Делегування медіаданих до внутрішнього аналітичного сервісу без порушення бінарної структури.
ФВ-О-05	Зворотний зв’язок	Автоматична ініціація HTTP-запиту (Webhook) на цільову адресу після отримання фінального вердикту.

Таблиця 2.4 – Нефункціональні вимоги до керуючого сервісу

ID	**Категорія**	**Метрика**	**Опис вимоги**
1	2	3	4
НФ-О-01	Масштабованість	≥ 6000 паралельних сесій	Здатність обробляти паралельні структури даних без блокування потоків.

Продовження таблиці 2.4

1	2	3	4
НФ-О-02	Продуктивність	≤ 200 мілісекунд	Час, витрачений на перевірку запиту та його передачу до сервісу комп’ютерного зору.
НФ-О-03	Безпека доступу	Точно 10 хвилин	Максимальний час життєздатності згенерованого криптографічного токена до його автоматичного анулювання.
НФ-О-04	Відмовостійкість	5 спроб із затримкою	Використання експоненційного згасання для доставки сповіщень при мережевих збоях зовнішньої системи.

2.3.3 Вимоги до сервісу комп’ютерного зору

Сервіс комп’ютерного зору (аналітичний компонент) виконує роль обчислювального ядра системи. Він позбавлений будь-яких функцій мережевої взаємодії із зовнішнім світом і зосереджений виключно на обробці математичних алгоритмів та роботі нейромереж. Функціональне призначення компонента наведено на рисунку 2.7.

Рисунок 2.7 – Зони відповідальності сервісу комп’ютерного зору

Функціональні вимоги цього компонента відповідають етапам нейромережевого аналізу, описаним у попередніх розділах (табл. 2.5).

Таблиця 2.5 – Функціональні вимоги до сервісу комп'ютерного зору

ID	**Назва вимоги**	**Детальний опис функціональності**
ФВ-А-01	Оптичне розпізнавання	Вилучення машинозчитуваної зони з фотографії документа та її конвертація в текстовий формат.
ФВ-А-02	Векторизація обличчя	Генерація 128-мірних ембедингів для обличчя з паспорта та фотографії користувача.
ФВ-А-03	Перевірка життєздатності	Аналіз динаміки зміни коефіцієнта EAR та дисперсії відносних відстаней для доведення живої присутності.
ФВ-А-04	Формування вердикту	Компіляція обчислених метрик подібності у стандартизований JSON-пакет із фінальним статусом перевірки.

Нефункціональні вимоги до сервісу аналітики стосуються точності нейромережевих моделей, швидкодії обчислень та суворої політики збереження даних (табл. 2.6).

Таблиця 2.6 – Нефункціональні вимоги до сервісу комп'ютерного зору

ID	**Категорія**	**Метрика / Обмеження**	**Опис вимоги**
1	2	3	4
НФ-А-01	Швидкодія (Інференс)	≤ 3500 мілісекунд	Сумарний час, необхідний на розпізнавання тексту, детекцію облич та векторне порівняння.

Продовження таблиці 2.6

1	2	3	4
НФ-А-02	Точність (FAR)	< 0,1%	Гранична ймовірність того, що система помилково розпізнає зловмисника як власника документа.
НФ-А-03	Ізоляція пам’яті	Негайне видалення	Абсолютне стирання всіх зображень та вилучених векторів з оперативної пам’яті відразу після формування вердикту.
НФ-А-04	Мережева ізоляція	Тільки внутрішня мережа	Відсутність публічних зовнішніх інтерфейсів; компонент повинен обробляти запити лише від керуючого сервісу.

Розподіл вимог на три окремі групи дозволяє забезпечити паралельну та незалежну розробку кожного мікросервісу, гарантуючи при цьому, що інтегрована система відповідатиме всім встановленим критеріям продуктивності та безпеки.

2.4 Розроблення протоколів обміну та безпеки даних

Розподілена природа розробленого комплексу вимагає використання надійних та стандартизованих протоколів обміну інформацією. Базовим архітектурним стилем взаємодії обрано підхід REST поверх безпечного протоколу передачі гіпертексту (HTTPS). Особливістю проєкту є необхідність одночасної передачі великих бінарних об’єктів

(фотографій високої роздільної здатності) та структурованих масивів числових координат ключових точок обличчя [44].

Для оптимізації мережевого трафіку обґрунтовано використання формату multipart/form-data. Альтернативний підхід, що передбачає кодування зображень у формат Base64 для передачі у стандартному тілі JSON-запиту, визнано неефективним. Математично доведено, що кодування Base64 збільшує об’єм переданих даних на третину. Якщо розмір вихідного бінарного файлу становить S_{bytes}, то розмір закодованого рядка S_{Base64} визначається за формулою:

$$S_{Base64} = 4 \cdot \left\lceil \frac{S_{bytes}}{3} \right\rceil \approx 1,33 \cdot S_{bytes}. \quad (2.10)$$

Уникнення цього надлишкового перетворення на клієнтському боці дозволяє суттєво зменшити час завантаження даних (T_{net}), що позитивно впливає на виконання загальних нефункціональних вимог системи.

Для захисту периметра та надійної ідентифікації сесій користувачів розроблено механізм генерації безпечних маркерів доступу (Secure Token Service). Після створення нової сесії керуючий сервіс генерує криптографічний токен T. Процес генерації базується на використанні криптографічно стійкого генератора псевдовипадкових чисел та алгоритмі хешування HMAC-SHA256 [26]. Формально структура токена описується рівняння:

$$T = Base64UrlEncode\left(HMAC - SHA256\left(ID_{session} \| Timestamp, \mathrm{Sec}\, retKey\right)\right). \quad (2.11)$$

Використання 256-бітного простору ключів забезпечує високий рівень інформаційної ентропії, що робить підбір токена методом повного перебору обчислювально неможливим завданням [45]. Кожному згенерованому маркеру жорстко прив’язується мітка часу створення (T_{create}). Впроваджено

механізм обмеження терміну дії (Time-to-Live), за яким токен вважається дійсним лише за виконання математичної нерівності:

$$T_{current} - T_{create} \le TTL_{\max}, \tag{2.12}$$

де $TTL_{\max}$ – максимально допустимий час активності, який становить 600 секунд.

Процес обміну інформацією та перевірки токенів під час взаємодії клієнтського застосунку з керуючим сервісом зображено на рисунку 2.8.

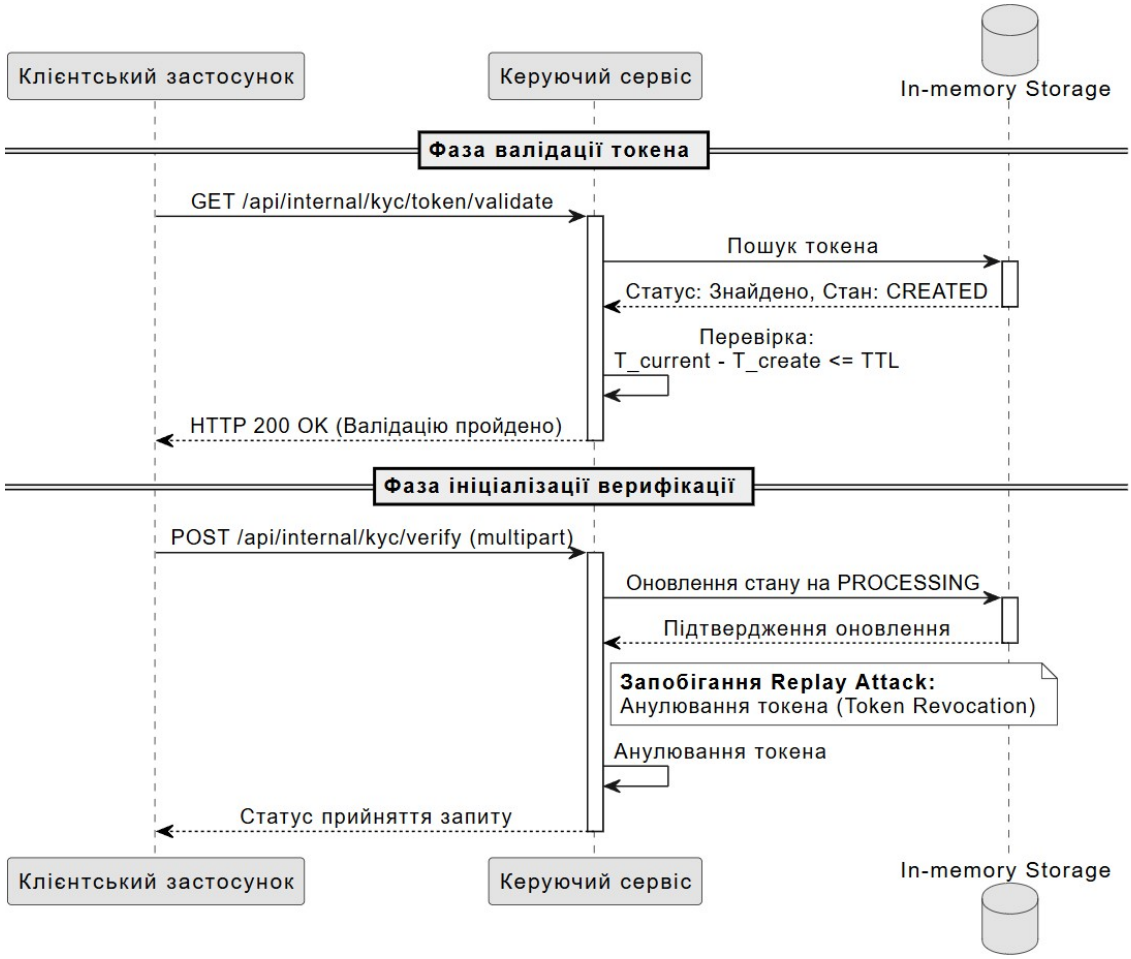

Рисунок 2.8 – Діаграма послідовності обміну даними та перевірки доступу

Крім того, в архітектурі реалізовано концепцію одноразовості: після переходу автомата сесії у стан обробки, відповідний токен невідворотно маркується як використаний. Цей механізм ефективно запобігає атакам типу

Replay Attack, коли зловмисник намагається перехопити і повторно відправити легітимний запит на верифікацію особи.

2.5 Проєктування логіки зворотного зв’язку

Завершальним етапом процесу верифікації є передача фінального результату до зовнішньої системи, яка виступила ініціатором перевірки. В архітектурі клієнт-серверних систем існує два базові підходи до отримання асинхронних результатів: періодичне опитування (Polling) та механізм зворотних викликів (Webhook) [27].

При використанні моделі опитування зовнішня система змушена періодично надсилати запити до керуючого сервісу, щоб дізнатися статус поточної сесії. Кількість таких мережевих повідомлень безпосередньо залежить від того, як довго триває загальне оброблення даних, та від частоти, з якою система перевіряє оновлення.

Оскільки нейромережевий аналіз потребує значних ресурсів, а в окремих випадках сесія може бути передана на ручну перевірку оператором, час очікування результату варіюється від кількох секунд до десятків хвилин. За таких умов модель постійного опитування створює велику кількість надлишкових запитів. Це призводить до неефективного використання потужностей центрального процесора керуючого сервісу та марного завантаження каналів зв’язку. Порівняльний аналіз цього підходу та альтернативної архітектури наведено у таблиці 2.7.

Для забезпечення високої масштабованості у проєкті реалізовано механізм HTTP-сповіщень (Webhook). На відміну від моделі опитування, тут напрямок зв’язку інвертується: керуючий сервіс запам’ятовує URL-адресу, надану зовнішньою системою, і самостійно ініціює асинхронний запит із результатами одразу після завершення обробки. Це дозволяє обмежитися лише одним вихідним повідомленням замість серії запитів.

Таблиця 2.7 – Порівняння механізмів отримання результатів

Критерій оцінювання	**Модель опитування (Polling)**	**Механізм зворотних викликів (Webhook)**
Кількість мережевих запитів	Велика (залежить від часу обробки)	Мінімальна (зазвичай 1 запит)
Витрати ресурсів сервера	Високі (постійні звернення до пам’яті)	Низькі (подійно-орієнтована архітектура)
Затримка отримання даних	Дорівнює половині інтервалу опитування	Відсутня (результат надсилається миттєво)
Складність реалізації	Низька (простий HTTP-клієнт)	Середня (потребує реалізації відкритих кінцевих точок на стороні клієнта)

Щоб гарантувати доставку результатів навіть у разі нестабільної мережі, в архітектуру закладено алгоритм експоненційного згасання (Exponential Backoff). Якщо сервер отримувача недоступний, керуючий сервіс здійснює повторні спроби, при цьому пауза між ними прогресивно зростає з кожним наступним кроком. Для запобігання масовим одночасним навантаженням на мережу (проблема Thundering Herd) до інтервалу очікування додається випадкове відхилення (джиттер). Логіку роботи цього механізму продемонстровано на рисунку 2.9.

Застосування даного підходу дозволяє перевести роботу системи у повністю подійно-орієнтований (Event-Driven) режим, що розвантажує обчислювальні потужності від обробки порожніх звернень та гарантує доставку результату перевірки зовнішній системі незалежно від тимчасових апаратних збоїв на стороні одержувача [46].

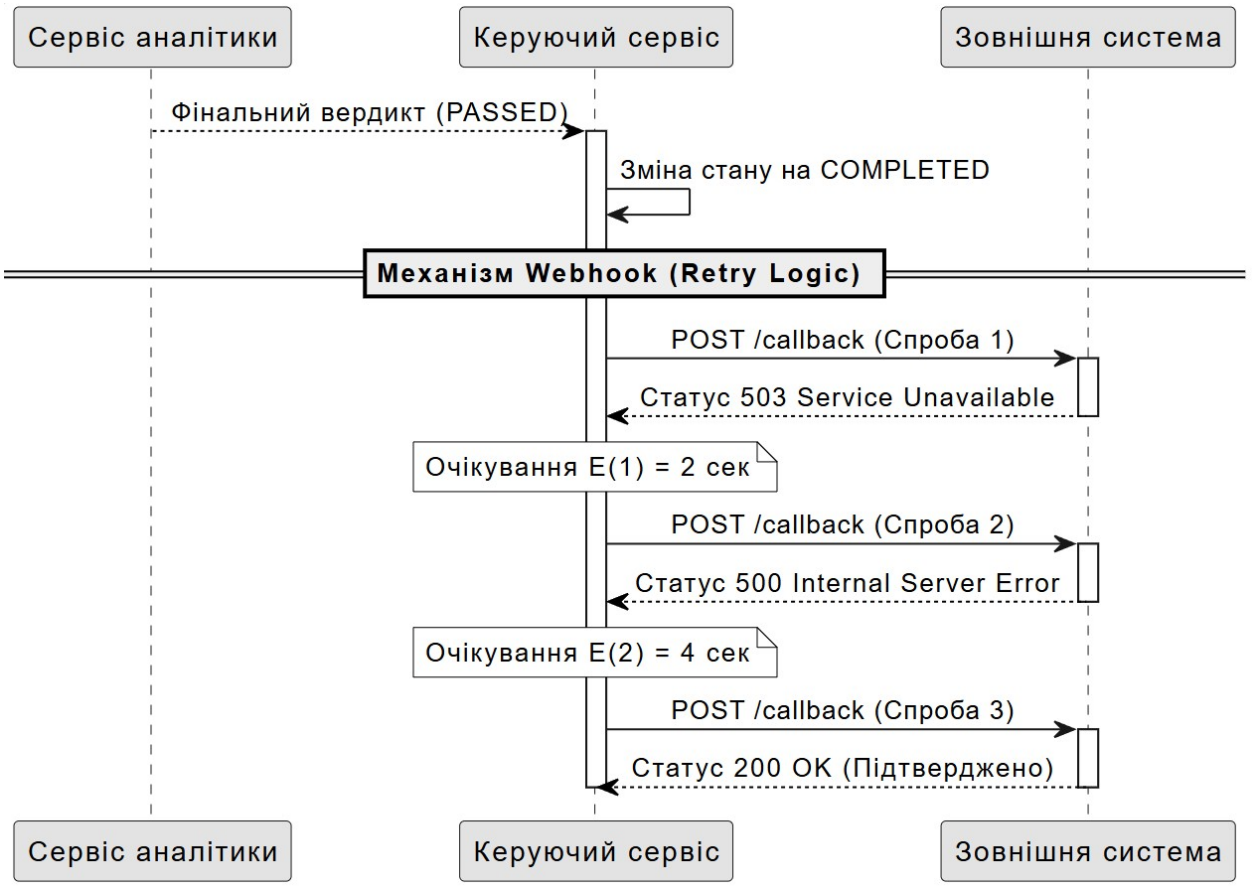

Рисунок 2.9 – Діаграма взаємодії з використанням механізму HTTP-сповіщень та повторних спроб

3 РОЗРОБЛЕННЯ СИСТЕМИ ТА ТЕСТУВАННЯ

3.1 Обґрунтування вибору інструментів розроблення

Розроблення мікросервісної системи віддаленого перевіряння користувачів дозволяє обрати оптимальний технологічний набір для кожного модуля для раціонального використання ресурсів і підвищення надійності. Першочерговим завданням є визначення технологій для керівного сервісу, сервісу комп’ютерного зору, клієнтського застосунку та імітаційної системи.

Керівний сервіс координує запити від зовнішніх систем, генерує сесії та маршрутизує дані. Для його реалізації обрано мову Java 21, оскільки її статична типізація мінімізує помилки, а віртуальні потоки (Virtual Threads) оптимізують паралельне оброблення запитів в оперативній пам'яті.

Базовим каркасом визначено Spring Boot 3, компоненти Spring Web якого спрощують створення API та відправлення зворотних викликів [47]. Завдяки впровадженню залежностей (Dependency Injection), архітектура залишається гнучкою для тестування. Під час проєктування також розглядалися платформи C# (.NET) та Node.js (NestJS), порівняння яких наведено в таблиці 3.1. Node.js поступається Java в обчислювальних завданнях через однопотокову модель, а C# має схожу продуктивність, проте Java володіє ширшою спільнотою та більшою кількістю готових бібліотек [48].

Сервіс комп’ютерного зору є ізольованим мікросервісом для ресурсомісткого аналізу медіаданих. Він виконує перевірку присутності реальної людини (liveness detection), порівняння облич та оптичне розпізнавання символів (OCR). Специфіка цих завдань зумовила вибір мови програмування Python, яка має розвинені бібліотеки для машинного навчання та матричних обчислень, що інтегрують низькорівневий код на C/C++ для максимальної швидкодії.

Таблиця 3.1 – Порівняння платформ для основного керуючого сервісу

Критерій	**Java (Spring Boot)**	**C# (.NET)**	**Node.js (NestJS)**
Продуктивність під навантаженням	Висока (багатопотоковість)	Висока (багатопотоковість)	Середня (однопотокова подієва модель)
Типізація	Строга статична	Строга статична	TypeScript (компілюється у динамічний JS)
Екосистема для мікросервісів	Дуже розвинена (Spring Cloud)	Розвинена	Розвинена
Швидкість розроблення	Середня	Середня	Висока

Для оброблення зображень та відеопотоків залучено бібліотеку OpenCV. Зчитування текстових даних із документів реалізовано за допомогою системи Tesseract OCR. Для порівняння векторних подань облич використано бібліотеку face-recognition, яка базується на переднавчених моделях машинного навчання та є стійкою до змін освітлення. При виборі вебплатформи для Python порівнювалися FastAPI, Flask та Django (табл. 3.2) [49, 50].

Для реалізації вузькоспеціалізованого сервісу обрано FastAPI через його високу швидкодію та підтримку асинхронності, на відміну від монолітного Django чи Flask з обмеженою асинхронністю. Клієнтський вебзастосунок розроблено на чистому JavaScript (Vanilla JS) з HTML5 та CSS3, що забезпечує стабільну роботу в мобільних браузерах, швидке завантаження та відсутність накладних витрат на Virtual DOM.

Таблиця 3.2 – Порівняння вебфреймворків для сервісу комп’ютерного зору

Критерій	**Flask**	**Django**	**FastAPI**
Архітектура	Мікрофреймворк	Монолітний	Мікрофреймворк
Асинхронність (Async/Await)	Обмежена підтримка	Підтримується частково	Вбудована повна підтримка
Швидкодія API	Середня	Низька (через надлишковість)	Дуже висока
Автогенерація документації	Відсутня	За допомогою сторонніх модулів	Вбудована (Swagger UI)

Взаємодія з камерою для захоплення біометричних даних реалізована через стандартний інтерфейс WebRTC, який підтримується всіма сучасними оглядачами. Використання важких каркасів є технічно та економічно недоцільним, оскільки інтерфейс системи складається з однієї динамічної сторінки, а порівняння чистого JavaScript із React та Vue.js наведено в таблиці 3.3.

Для об’єднання мікросервісів в єдину інфраструктуру та локального запуску застосовано технології Docker і Docker Compose. Кожен мікросервіс описується власним Dockerfile. Імітаційну зовнішню платформу (Mock Service) розроблено на Python із використанням локальної бази даних SQLite, що спрощує адміністрування. Для тестування механізму зворотних викликів (webhook) у локальних умовах використано утиліту безпечного тунелювання Ngrok, яка перенаправляє публічний трафік на локальний проксі-сервер Nginx. Загальну архітектуру технологічного стека та взаємодію компонентів представлено на діаграмі компонентів (рис. 3.1).

Таблиця 3.3 – Порівняння інструментів для розроблення клієнтського застосунку

Критерій	**Vanilla JS (без фреймворків)**	**React**	**Vue.js**
Розмір кінцевого застосунку	Мінімальний (кілька кілобайтів)	Великий (потребує React-DOM)	Середній
Складність збирання проєкту	Не потребує збирання	Висока (Webpack/Vite)	Висока (Vite)
Доступ до WebRTC (камера)	Прямий через API браузера	Через життєві цикли компонентів	Через директиви та виклики

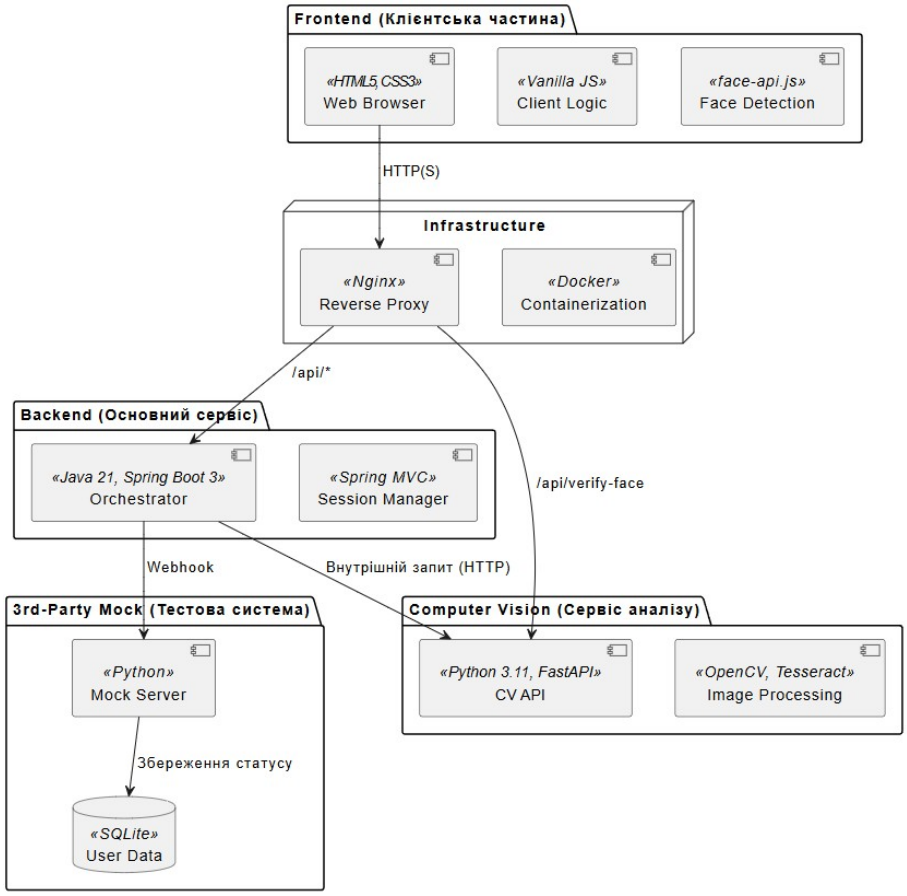

Рисунок 3.1 – Діаграма компонентів технологічного стека розробленої системи

3.2 Розроблення сервісу керування верифікацією

Взаємодія із зовнішніми системами та клієнтським застосунком реалізована через програмний інтерфейс (API) у класі KycController. Цей контролер містить кінцеві точки для створення, отримання статусу та скасування сесії верифікації. Документування інтерфейсу виконано за стандартом OpenAPI.

Кожен запит проходить обов'язкове перевіряння структури даних. Захист кінцевих точок забезпечується шляхом перевіряння токена доступу у заголовках HTTP-запитів. Логіку первинного оброблення заголовка авторизації наведено у лістингу 3.1.

Лістинг 3.1 Метод вилучення та первинної перевіряння токена доступу:

```
private String extractBearerToken(String authorizationHeader) {
    if (authorizationHeader == null || authorizationHeader.isBlank()) {
        throw new ResponseStatusException(HttpStatus.UNAUTHORIZED,
«Missing Authorization header»);
    }
    String prefix = «Bearer»;
    if (!authorizationHeader.regionMatches(true, 0, prefix, 0, prefix.length()))
{
        throw new ResponseStatusException(HttpStatus.BAD_REQUEST,
«Invalid Authorization scheme»);
    }
    String token = authorizationHeader.substring(prefix.length()).trim();
    if (token.isEmpty()) {
        throw new ResponseStatusException(HttpStatus.BAD_REQUEST,
«Bearer token is empty»);
    }
    return token;
```

}

Алгоритм вилучає токен і перевіряє його відповідність схемі Bearer. При виявленні невідповідностей генерується виняткова ситуація ResponseStatusException з відповідним кодом помилки, що блокує подальше виконання запиту та захищає бізнес-логіку від некоректних звернень.

Для керування тимчасовими ключами у пам’яті сервера розроблено компонент TokenManager. Токени генеруються за допомогою криптографічно стійкого генератора SecureRandom та кодуються у рядок стандарту Base64. Кожен токен містить інформацію про термін дії та поточний статус. Життєвий цикл токена представлено на рисунку 3.2.

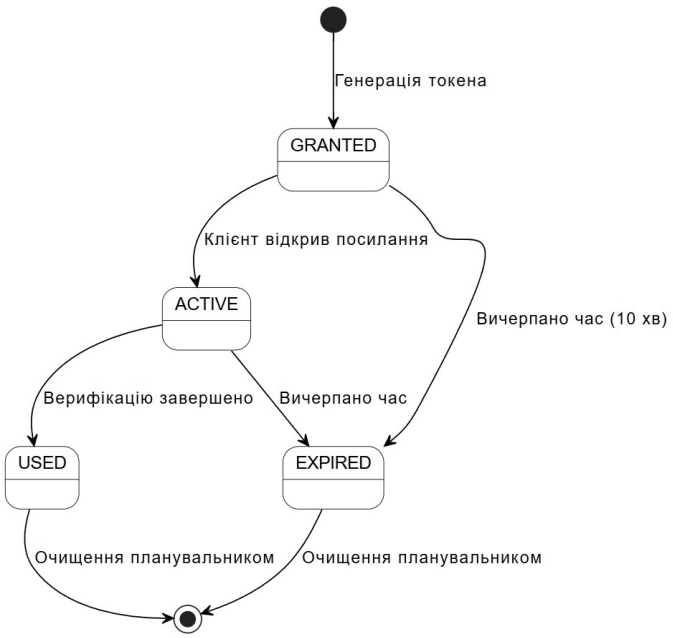

Рисунок 3.2 – Діаграма станів життєвого циклу токена доступу

Термін дії токена обмежено 10 хвилинами для запобігання повторному використанню посилань сторонніми особами. Видалення прострочених ключів з оперативної пам’яті виконується періодично за допомогою планувальника завдань з анотацією @Scheduled.

Клас VerificationService керує життєвим циклом сесій верифікації. Під час ініціалізації створюється новий токен та унікальний ідентифікатор сесії

(UUID). Дані зберігаються у потокобезпечній структурі ConcurrentHashMap, що запобігає конфліктам у багатопотоковому середовищі. Послідовність виконання операцій при ініціалізації сесії зображено на рисунку 3.3.

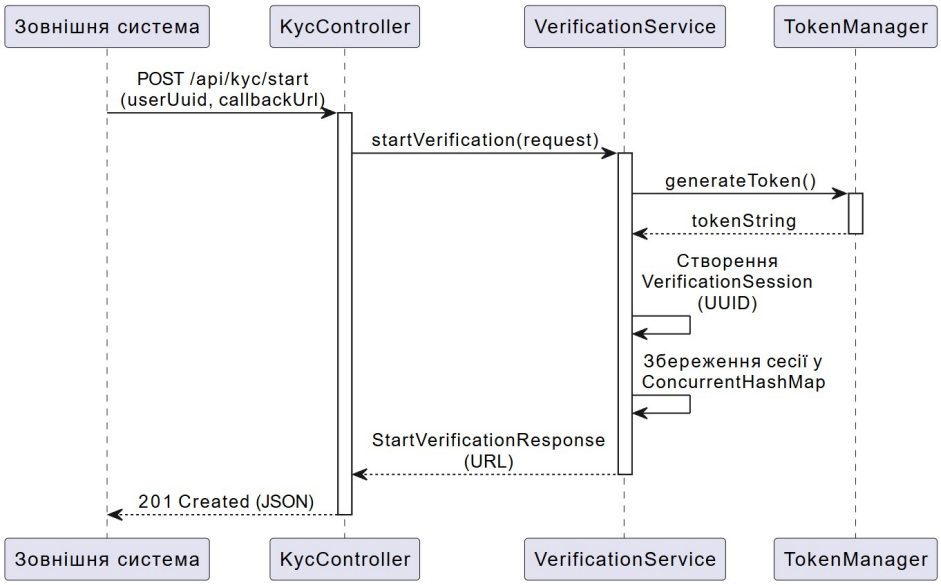

Рисунок 3.3 – Діаграма послідовності ініціалізації сесії верифікації

Основний Java-сервіс не виконує безпосереднього аналізу зображень, а передає медіадані до мікросервісу комп'ютерного зору за допомогою компонента PythonProxyClient. Для реалізації асинхронної неблокувальної взаємодії використано інструмент WebClient з екосистеми Spring WebFlux. Передавання файлів та метаданих здійснюється у форматі multipart/form-data. Програмний код формування запиту наведено у лістингу 3.2.

Лістинг 3.2 Формування складеного запиту до сервісу комп'ютерного зору:

```
public ProxyResponse forwardVerifyFace(String token, VerifyFaceRequest req) {
    MultipartBodyBuilder mb = new MultipartBodyBuilder();
    addFilePart(mb, «photo1», req.photo1());
    addFilePart(mb, «photo2», req.photo2());
```

```
        addFilePart(mb, «photo3», req.photo3());
        addFilePart(mb, «passport_photo», req.passportPhoto());

        mb.part(«lmk1», toJson(req.lmk1()));
        mb.part(«lmk2», toJson(req.lmk2()));
        mb.part(«lmk3», toJson(req.lmk3()));

        ResponseEntity<String> entity = webClient.post()
            .uri(«/api/verify-face»)
            .header(HttpHeaders.AUTHORIZATION, «Bearer» + token)
            .contentType(MediaType.MULTIPART_FORM_DATA)
            .body(BodyInserters.fromMultipartData(mb.build()))
            .retrieve()
            .toEntity(String.class)
            .block(Duration.ofSeconds(70));
        return new ProxyResponse(entity.getStatusCode(), entity.getBody(),
entity.getHeaders().getContentType());
    }
```

Використання реактивного клієнта запобігає блокуванню потоків вебсервера під час очікування результатів оброблення ресурсомістких завдань на стороні сервісу комп’ютерного зору. Структуру взаємодії класів серверної частини представлено на рисунку 3.4.

Оповіщення зовнішньої системи про результати перевірки виконується асинхронно через механізм зворотних викликів (webhook), що усуває потребу в постійному опитуванні сервера з боку замовника. Код відправлення сповіщення представлено у лістингу 3.3.

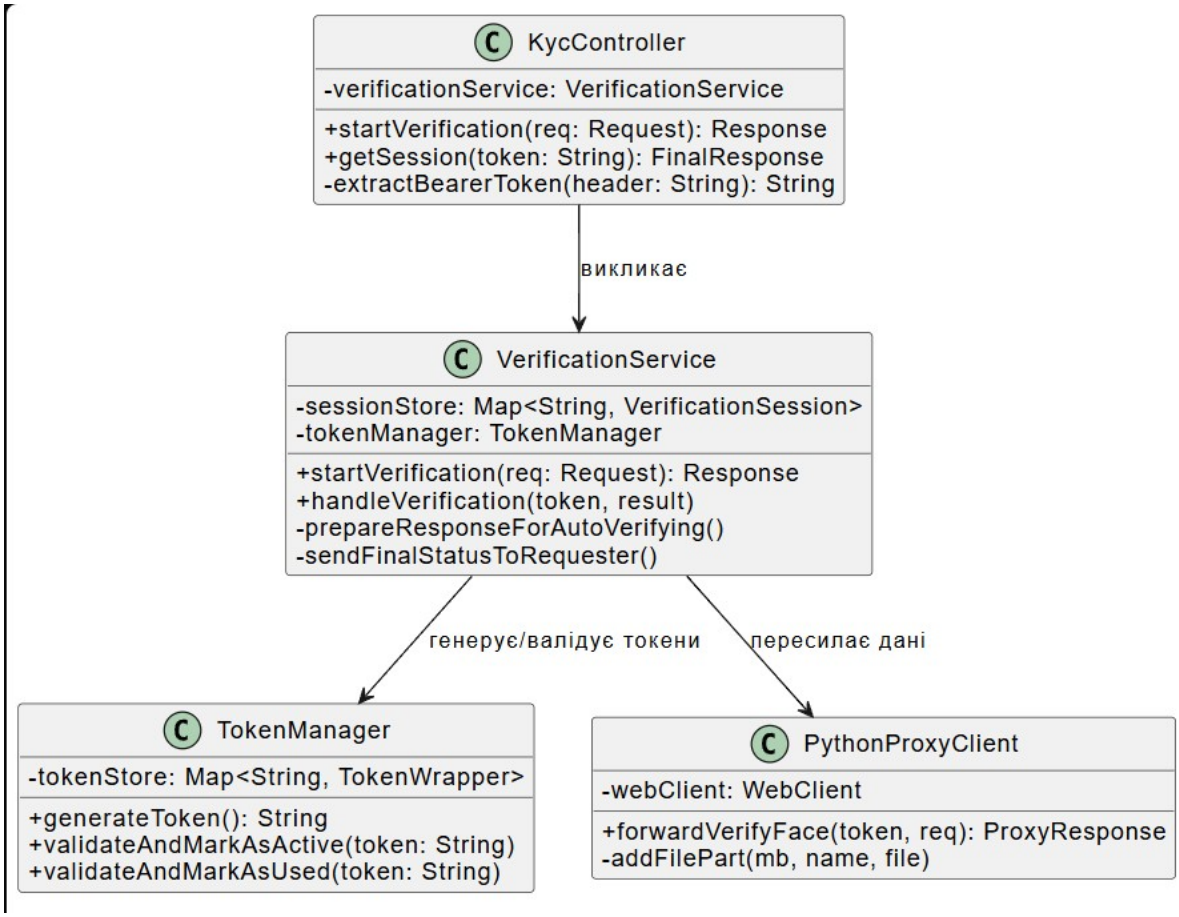

Рисунок 3.4 – Діаграма класів основного сервісу керування

Лістинг 3.3 Відправлення результату верифікації через механізм зворотних викликів:

```
private void sendFinalStatusToRequester(FinalResponse finalResponse, VerificationSession session) {
    WebClient client = WebClient.create(session.getCallbackUrl());

    client.post()
        .contentType(MediaType.APPLICATION_JSON)
        .bodyValue(finalResponse)
        .retrieve()
        .toBodilessEntity()
        .block();
}
```

3.3 Реалізація сервісу комп’ютерного зору

Сервіс комп’ютерного зору є ізольованим модулем, який виконує обчислювально складні завдання системи: нейромережевий аналіз мультимедійних даних, отриманих від клієнта. Виокремлення цієї логіки в мікросервіс дозволяє ефективно масштабувати систему при пікових навантаженнях. Реалізацію виконано мовою Python, що забезпечує інтеграцію низькорівневих математичних бібліотек для швидкого обчислення біометричних векторів ознак.

Взаємодія з керівним сервісом відбувається через API за допомогою HTTP-методу POST. Для передавання бінарних файлів і текстових метаданих застосовано формат multipart/form-data. Кінцева точка мікрокаркаса FastAPI розпаковує параметри, перевіряє типи даних та наявність токена авторизації у заголовках. Програмну конфігурацію кінцевої точки наведено у лістингу 3.4.

Лістинг 3.4 Оголошення кінцевої точки для прийому медіаданих:

```
@app.post(«/api/verify-face»)
async def verify_face(
    photo1: UploadFile = File(...),
    photo2: UploadFile = File(...),
    photo3: UploadFile = File(...),
    lmk1: str = Form(...),
    lmk2: str = Form(...),
    lmk3: str = Form(...),
    passport_photo: Optional[UploadFile] = File(None),
    authorization: Optional[str] = Header(default=None),
):
  normalized_authorization = authorization.strip() if authorization else None
```

```
    bearer_token = extract_bearer_token(normalized_authorization)
    # Подальша логіка декодування та аналізу
```

Функція приймає три обов’язкові зображення обличчя, координати контрольних точок у форматі JSON та необов’язкове фото документа. Усі зображення перетворюються на матриці RGB для подальшого оброблення та передавання до модуля визначення присутності людини.

Цей модуль захищає систему від використання статичних фотографій чи відеозаписів. Замість ресурсомісткого виявлення обличчя на сервері, алгоритм використовує заздалегідь обчислені клієнтською частиною набори контрольних антропометричних точок. Це знижує навантаження на обчислювальні вузли та прискорює оброблення запитів. Алгоритм послідовно перевіряє рух голови, зміну паралакса та узгодженість напрямку обертання. Реалізацію функції перевірки псевдотривимірного паралакса наведено у лістингу 3.5.

Лістинг 3.5 Алгоритм перевірки псевдотривимірного паралакса обличчя:

```
def _check_parallax(landmarks: List[dict]) -> Dict[str, Any]:
    ratios = []
    for lm in landmarks:
        nose = (lm[«nose»][«x»], lm[«nose»][«y»])
        left_eye = (lm[«leftEye»][«x»], lm[«leftEye»][«y»])
        right_eye = (lm[«rightEye»][«x»], lm[«rightEye»][«y»])

        d_left = _dist(nose, left_eye)
        d_right = _dist(nose, right_eye)

        if d_left != 0 and d_right != 0:
            ratios.append(d_left / d_right)
```

delta = max(ratios) - min(ratios) if len(ratios) >= 2 else 0

if delta < 0.05:

return {«ok»: False, «status»: «fail», «reason»: «No parallax change»}

return {«ok»: True, «status»: «ok», «reason»: «Parallax change detected»}

Алгоритм розраховує відношення евклідових відстаней від кінчика носа до лівого та правого ока на трьох послідовних кадрах. При реальному повороті голови ці пропорції змінюються через зміщення ракурсу, тоді як для плоских зображень вони залишаються сталими. Якщо різниця відношень менша за поріг (0,05), система класифікує запит як спробу шахрайства. Схему конвеєра аналізування зображень наведено на діаграмі діяльності (рис. 3.5).

Після підтвердження присутності людини оцінюється якість документа: освітленість – за середньою яскравістю, а розмитість – за допомогою оператора Лапласа (Laplacian variance) у відтінках сірого. Зображення з дисперсією менше 50 відхиляються.

Додатково контролюються геометричні пропорції документа (співвідношення сторін від 1,20 до 1,70) та площа обличчя (не більше 40% кадру). Також аналізується колірний профіль у просторі HSV для виявлення відтінків шкіри, а наявність машинозчитуваної зони (MRZ) перевіряється оцінюванням щільності горизонтальних градієнтів у нижній третині зображення за допомогою фільтра Собеля. Метод формування еталонного вектора ознак наведено у лістингу 3.6.

Лістинг 3.6 Метод формування еталонного вектора ознак обличчя:

def build_reference_embedding(images: list) -> np.ndarray:

valid_encodings = []

for idx, img in enumerate(images, start=1):

```
            try:
                enc = extract_face_encoding(img)
                valid_encodings.append(enc)
            except ValueError as e:
                if «Multiple faces» in str(e):
                    raise ValueError(f»Liveness frame {idx}: {e}»)

        if len(valid_encodings) < 2:
            raise ValueError(«Could not build reference embedding: not enough valid frames»)

        return np.mean(valid_encodings, axis=0)
```

Метод у лістингу 3.6 створює усереднений вектор ознак обличчя. Бібліотека face_recognition перетворює зображення на унікальний масив із 128 чисел із плаваючою комою. Алгоритм ігнорує кадри без обличчя та перериває процес, якщо виявлено кілька осіб. Розрахунок математичного сподівання векторів забезпечує точність цифрового зліпка, для чого необхідно успішно опрацювати принаймні два кадри з трьох.

Далі вектор ознак із фото паспорта порівнюється з еталонним шляхом розрахунку евклідової відстані у 128-вимірному просторі. Для класифікації застосовано систему порогів: значення до 0,46 означають високу подібність (автоматичне схвалення), від 0,46 до 0,55 – надсилання на ручну перевірку, а понад 0,55 – автоматичне відхилення через невідповідність біометрії.

Підсистема OCR вилучає структуровані текстові дані з машинозчитуваної зони (MRZ). Вирізане зображення проходить попереднє оброблення для підвищення контрастності: перетворення у відтінки сірого, вирівнювання гістограми, фільтрацію Гаусса та локальне бінаризування. Для локалізації зони застосовується морфологічне перетворення Blackhat. Розпізнавання двома незалежними рушіями наведено у лістингу 3.7.

Рисунок 3.5 – Діаграма діяльності процесу нейромережевого аналізу зображень

Лістинг 3.7 Логіка виконання розпізнавання двома рушіями:

```
def extract_passport_dob(passport_img: np.ndarray) -> dict:
    fallback_mrz, _ = _extract_mrz_roi(passport_img)
    mrz_img = fallback_mrz
```

```
    raw_text = None
    try:
        raw_text = _postprocess_mrz_text(_ocr_pytesseract(_preprocess_mrz(mrz_img)))
    except Exception as e:
        logger.warning(«[OCR] pytesseract failed: %s», e)
    if not raw_text:
        try:
            raw_text = _ocr_easyocr(mrz_img)
        except Exception as e:
            logger.warning(«[OCR] easyocr failed: %s», e)
    mrz_parsed = parse_td3_mrz(raw_text or «»)
    return {«dob»: _parse_dob(raw_text or «»), «mrz»: mrz_parsed}
```

Для забезпечення високої надійності основним інструментом обрано pytesseract, придатний для стандартизованих шрифтів. У разі виникнення помилок система автоматично перемикається на резервну нейромережеву модель easyocr. Вилучений текст нормалізується для усунення типових замін символів (наприклад, плутанини між «O» та «0») і передається до аналізатора.

Аналізатор машинозчитуваної зони перевіряє структуру за стандартом ICAO 9303 (формат TD3). Для номера документа, дати народження та терміну дії розраховується контрольна сума за алгоритмом із ваговими коефіцієнтами (7, 3, 1). Невідповідність контрольного числа призводить до визнання даних невідповідними. Сформований пакет результатів асинхронно надсилається на керівний сервіс за допомогою клієнта httpx із додаванням оригінального токена авторизації для підтвердження легітимності з’єднання.

3.4 Розроблення сервісу для захоплення зображень

Основою візуальної структури сторінки верифікування є динамічний контейнер із елементом відеотрансляції та графічними напрямними. Для відтворення потоку з камери використовується нативний тег HTML5 <video>, налаштований на автоматичне відтворення без звуку. Поверх відео накладається напівпрозорий шар, що динамічно змінює форму відповідно до поточного кроку: відображає овал для сканування обличчя або прямокутник для паспорта. Ця маска допомагає користувачеві правильно розташувати об’єкт у кадрі. Схематичне відображення інтерфейсу початкового екрана наведено на рисунку 3.6.

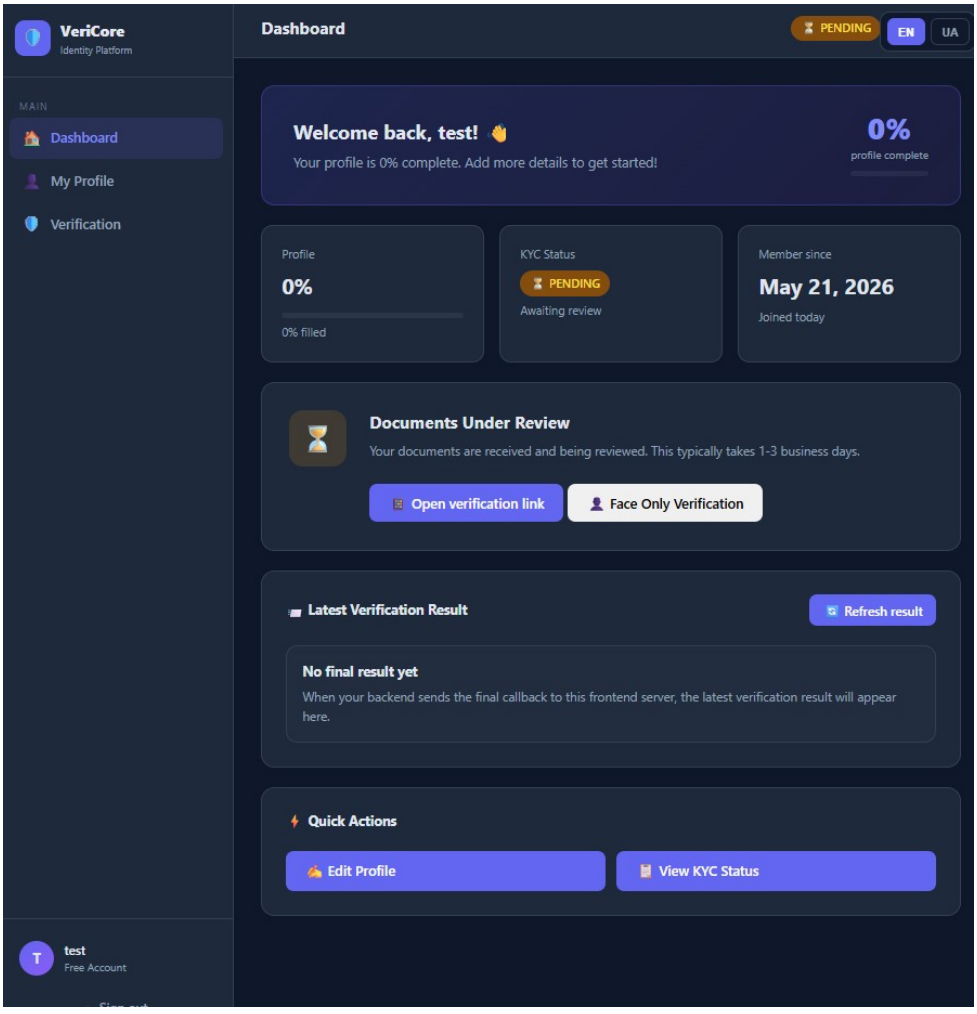

Рисунок 3.6 – Схема інтерфейсу початкового екрана клієнтського застосунку

Доступ до камери пристрою реалізовано через інтерфейс WebRTC методом navigator.mediaDevices.getUserMedia(). Для сумісності з різними типами об’єктивів розроблено алгоритм поступового зниження вимог до роздільної здатності: спочатку запитується формат Full HD (1920×1080) для високої деталізації, а у разі його непідтримування апаратним забезпеченням – стандартний HD-формат. Цей процес вимагає обов’язкового надання дозволу від користувача. Послідовність ініціалізування пристроїв захоплення зображень наведено на діаграмі (рис. 3.7).

Для оцінювання якості матеріалів до відправлення на сервер у вебоглядач інтегровано бібліотеку face-api.js. Завдяки апаратному прискоренню WebGL нейромережеві обчислення виконуються без перевантаження центрального процесора. Моделі завантажуються асинхронно та зберігаються в пам’яті для швидкого виявлення обличчя та надання користувачеві підказок щодо позиціонування голови в режимі реального часу.

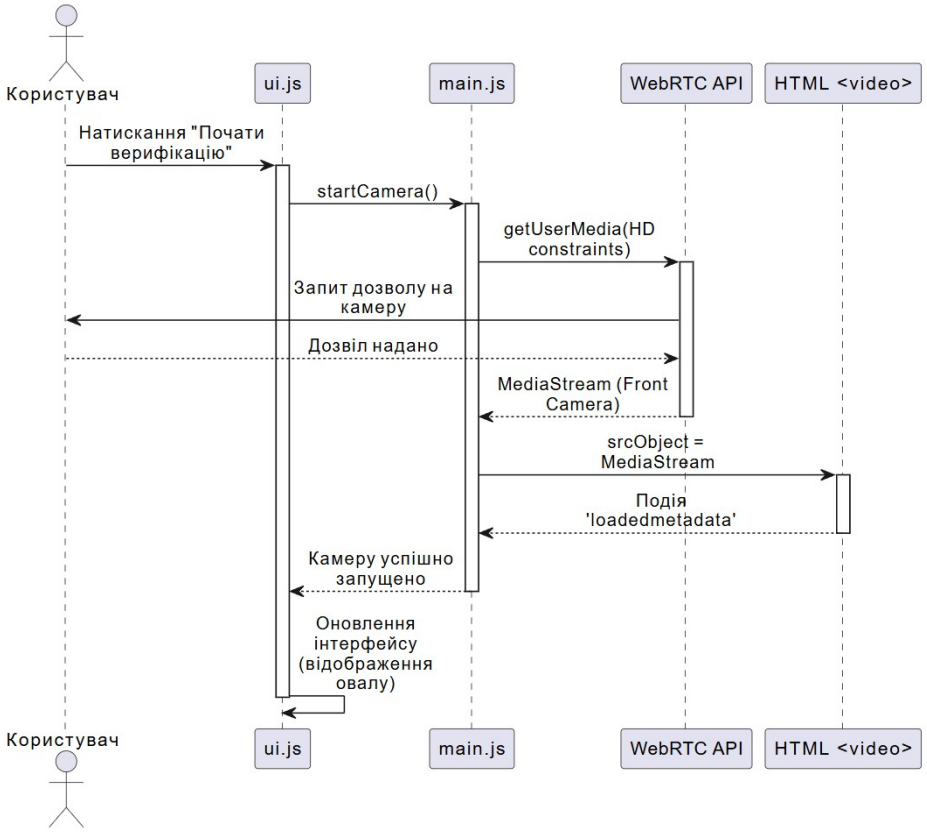

Рисунок 3.7 – Діаграма послідовності ініціалізації медіапристроїв

Процедура виявлення обличчя запускається періодично. Кадри проходять геометричне зіставлення, під час якого обчислюється площа обличчя відносно меж напрямного овалу та контролюється кут нахилу голови. Якщо пропорції відповідають обмеженням, рамка змінює колір на зелений, сигналізуючи про готовність до знімання (рис. 3.8).

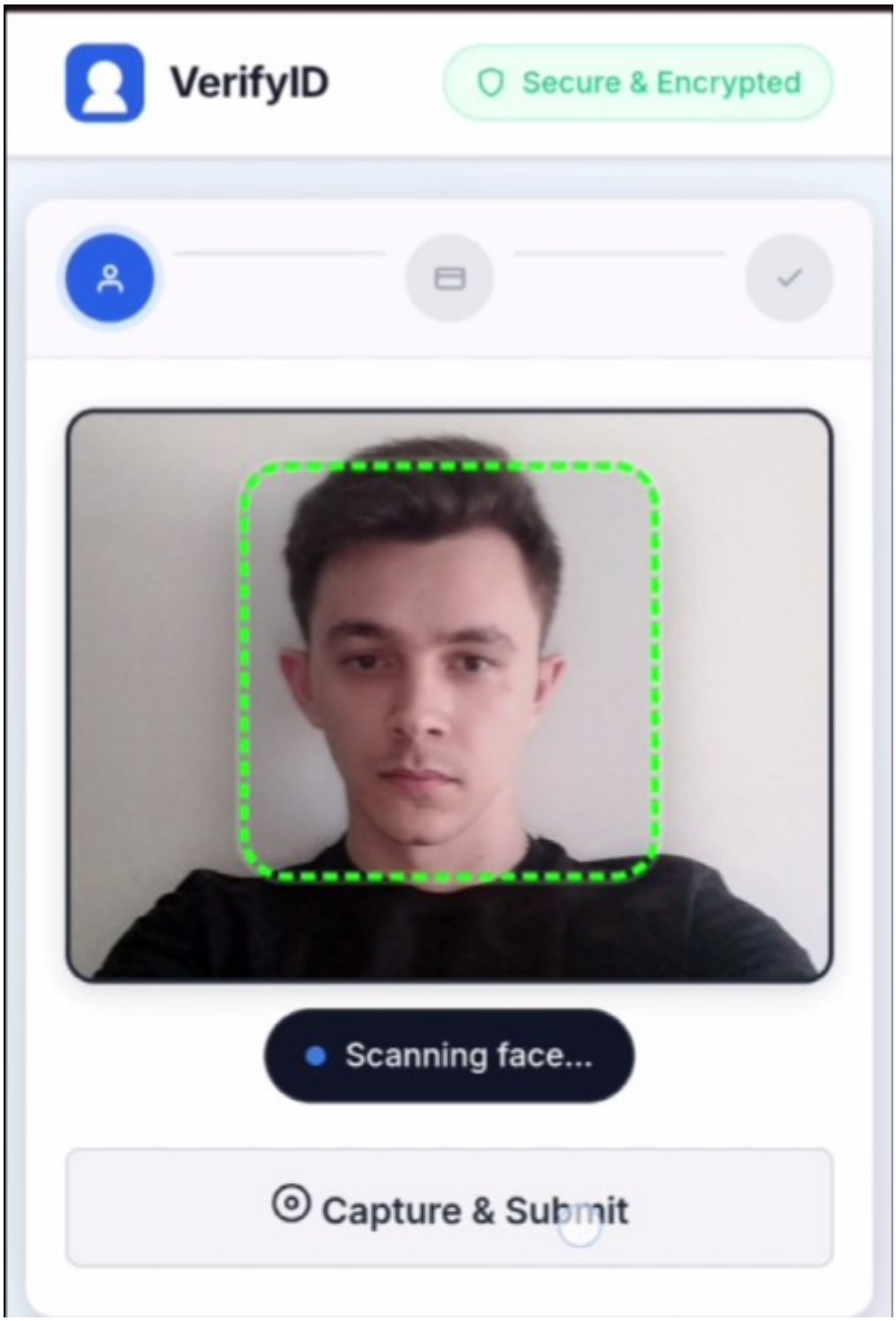

Рисунок 3.8 – Схема інтерфейсу під час успішного виявлення обличчя

Розрахунок геометричних відхилень виконується спеціалізованою функцією клієнтської частини. При виявленні відхилень від порогових конфігурацій процес переривається з виведенням підказки для коригування положення голови. Реалізацію перевіряння пропорцій обличчя наведено у лістингу 3.8.

Лістинг 3.8 Перевіряння пропорцій виявленого обличчя у кадрі:

```
function evaluateFaceGuideDetection(faceCanvas, detection) {
  const box = detection.detection.box;
  const frameWidth = faceCanvas.width;
  const frameHeight = faceCanvas.height;

  const widthFill = box.width / frameWidth;
  const heightFill = box.height / frameHeight;

  if (widthFill < FACE_DETECTION_CONFIG.minWidthFill || heightFill < FACE_DETECTION_CONFIG.minHeightFill) {
    return { ok: false, reason: «Будь ласка, наблизьте обличчя» };
  }
  if (widthFill > FACE_DETECTION_CONFIG.maxWidthFill || heightFill > FACE_DETECTION_CONFIG.maxHeightFill) {
    return { ok: false, reason: «Будь ласка, відсуньте пристрій далі» };
  }

  return { ok: true, reason: «», canvas: faceCanvas, landmarks: detection.landmarks };
}
```

Наведений алгоритм порівнює відносні розміри виявленого обличчя з нормалізованими коефіцієнтами, що забезпечує коректну роботу незалежно

від роздільної здатності камери чи характеристик екрана. Попереднє фільтрування невдалих кадрів знижує ймовірність відхилення запиту на серверній частині. Після вирівнювання обличчя запускається автоматизоване знімання трьох послідовних кадрів, де замість цифрового таймера застосовано трисекундну анімацію заповнення контуру напрямної рамки. При зникненні обличчя з кадру процес скасовується, а успішний зворотний відлік ініціює кадрування поточного буфера за допомогою функції у лістингу 3.9.

Лістинг 3.9 Алгоритм просторового кадрування зображення з урахуванням масштабу:

```
function cropFromFrame(video, frameElement) {
  const videoRect = video.getBoundingClientRect();
  const frameRect = frameElement.getBoundingClientRect();
  const layout = getObjectFitCoverLayout(video);
  const frameLeft_rel = frameRect.left - videoRect.left;
  const frameTop_rel  = frameRect.top  - videoRect.top;
  const topLeft = cssPxToVideo(frameLeft_rel, frameTop_rel, layout);

  let cropX = topLeft.x;
  let cropY = topLeft.y;
  let cropW = frameRect.width  / layout.scale;
  let cropH = frameRect.height / layout.scale;

  if (isVideoMirrored()) {
    cropX = video.videoWidth - (cropX + cropW);
  }

  const canvas = document.createElement(«canvas»);
  canvas.width = Math.round(cropW);
```

```
    canvas.height = Math.round(cropH);
    const ctx = canvas.getContext(«2d»);
    // Малювання кадру на полотно з урахуванням зсуву
    ctx.drawImage(video, cropX, cropY, cropW, cropH, 0, 0, canvas.width, canvas.height);
    return canvas;
  }
```

Функція обчислює масштабний коефіцієнт та зміщення, переводить координати рамки у фізичні координати камери, враховує дзеркальне відображення для фронтального модуля та копіює фрагмент на приховане полотно для генерування JPEG-файлу.

Для отримання знімка документа застосунок намагається перемикнути вебоглядач на тилову камеру пристрою. Алгоритм перебирає доступні медіасенсори, аналізуючи ідентифікатори на наявність ключових слів («back», «environment»). Якщо автоматичний вибір неможливий, залишається активний модуль, і користувачеві пропонується зафіксувати документ вручну. Керування зміною кроків перевіряння покладено на централізований менеджер станів інтерфейсу, який відображає прогрес та керує видимістю графічних компонентів. Для зчитування паспорта накладається прямокутна рамка, що допомагає правильно позиціонувати документ у полі зору камери. Схему інтерфейсу зчитування документа наведено на рисунку 3.9.

Виявлення документа в кадрі на клієнтській частині реалізовано без залучення важких нейромережевих моделей безпосередньо у вебоглядачі. Замість цього застосовано оцінювання різкості за допомогою дисперсії Лапласа, аналізування щільності вертикальних і горизонтальних градієнтів та пошуку чергувань світлих і темних смуг машинозчитуваної зони. Після фіксування стабільного положення документа протягом кількох кадрів виконується автоматичне знімання.

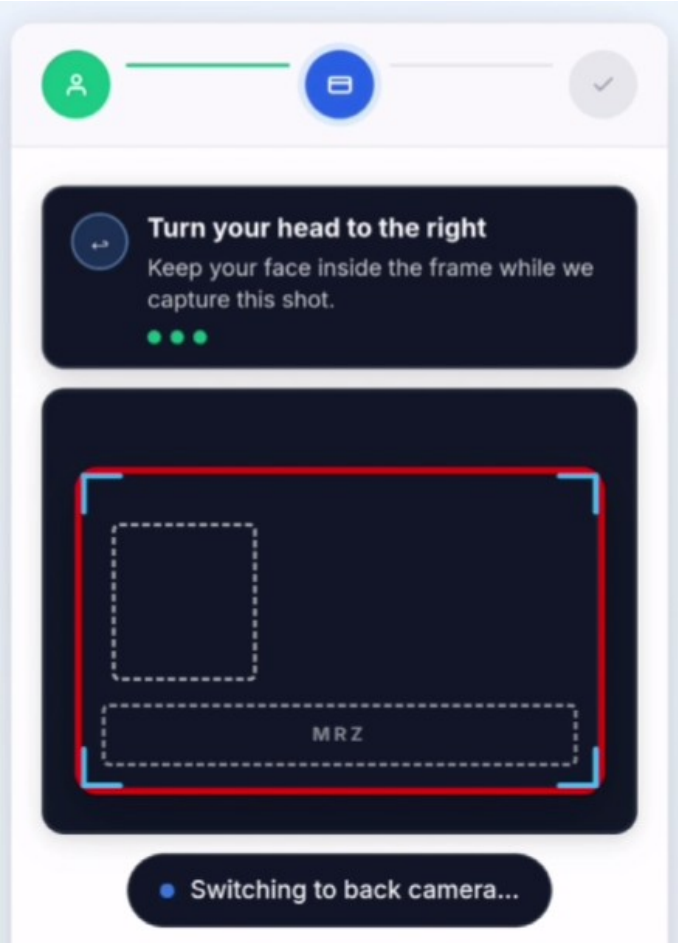

Рисунок 3.9 – Схема інтерфейсу користувача під час виявлення документа

Надсилання сформованого пакета даних на сервер здійснюється асинхронним HTTP-запитом, реалізованим у лістингу 3.10.

Лістинг 3.10 Формування та надсилання зібраних даних на оркестратор:

```
async function submitVerificationPayload({ faceCanvases, landmarks, passportCanvas }) {
  const formData = new FormData();

  for (let i = 0; i < 3; i++) {
    const photoBlob = await canvasToJpegBlob(faceCanvases[i]);
    formData.append(`photo${i + 1}`, photoBlob, `face_${i + 1}.jpg`);
    formData.append(`lmk${i + 1}`, JSON.stringify(landmarks[i] || null));
```

```
  }
  if (passportCanvas) {
   const passportBlob = await canvasToJpegBlob(passportCanvas);
   formData.append(«passport_photo», passportBlob, «passport.jpg»);
  }

  const response = await fetch(«/api/internal/kyc/verify», {
   method: «POST»,
   headers: { Authorization: window.getAuthorizationHeaderValue() },
   body: formData,
  });
  return { accepted: response.status === 202, statusCode: response.status };
 }
```

Кожне зображення конвертується у бінарний об’єкт (Blob) та додається до структури FormData разом із координатами контрольних точок. Маршрутизування через проксі-сервер Nginx усуває обмеження політики спільного походження (CORS). Після успішного оброблення користувач перенаправляється на фінальний екран (рис. 3.10).

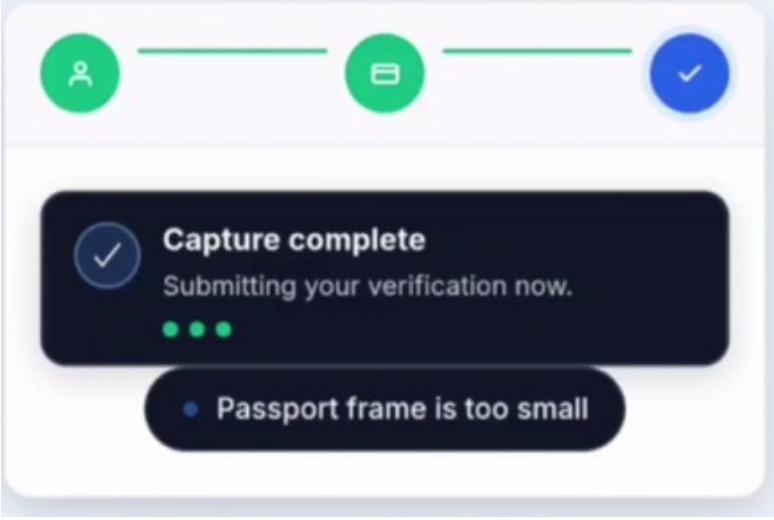

Рисунок 3.10 – Схема екрана успішного завершення процедури захоплення даних

3.5 Інтеграція компонентів системи

Для уніфікації середовищ виконання та спрощення процесу розгортання системи застосовано технологію контейнеризації Docker. Це дозволяє упакувати кожен мікросервіс разом із його залежностями в ізольований стандартизований образ, що усуває проблеми несумісності версій програмного забезпечення при перенесенні між різними апаратними платформами. Для автоматизованого оркестрування контейнерів у межах одного сервера використано інструмент Docker Compose, який дозволяє декларативно описати інфраструктуру в одному файлі конфігурації. Фрагмент цього файлу наведено у лістингу 3.11.

Лістинг 3.11 Фрагмент конфігурації оркестрування контейнерів:

```
services:
  kyc-orchestration-service:
    build:
      context: ./kyc-orchestration-service
    environment:
      - KYC_FRONTEND_BASE_URL=${PUBLIC_APP_URL:-http://localhost}
      - PYTHON_CV_HOST=kyc-cv-service
    networks:
      - kyc-network

  kyc-cv-service:
    build:
      context: ./kyc-cv-service
    environment:
      - JAVA_BACKEND_HOST=kyc-orchestration-service
    networks:
```

```
    - kyc-network
  depends_on:
    - kyc-orchestration-service
```

Конфігурація об’єднує основний керівний сервіс та модуль комп’ютерного зору в ізольовану мостову мережу kyc-network. Взаємодія між контейнерами здійснюється за допомогою внутрішніх доменних імен. Передавання параметрів підключення реалізовано через змінні оточення, що дозволяє змінювати налаштування без перекомпілювання коду. Директива depends_on координує послідовність запуску, ініціалізуючи сервіс аналізування зображень лише після старту Java-оркестратора.

Для усунення обмежень політики спільного походження (CORS), що виникають при взаємодії клієнтської частини з мікросервісами, які функціонують на різних портах, впроваджено зворотний проксі-сервер Nginx. Він приймає зовнішні HTTP-запити на порту 80 та маршрутизує їх до відповідних внутрішніх контейнерів. Конфігурацію маршрутизування наведено у лістингу 3.12.

Лістинг 3.12 Конфігурація зворотного проксі-сервера для маршрутизації:

```
server {
    listen 80;
    server_name localhost;

    # Маршрутизація до сервісу комп’ютерного зору
    location /api/verify-face {
        proxy_pass http://kyc-cv-service:8000/api/verify-face;
        client_max_body_size 50M;
    }
```

```
    # Маршрутизація до основного оркестратора
    location /api/ {
        proxy_pass http://kyc-orchestration-service:8080/api/;
    }

    # Роздача статичних файлів клієнтського застосунку
    location / {
        root /usr/share/nginx/html;
        index index.html;
        try_files $uri $uri/ /index.html;
    }
}
```

Директива client_max_body_size 50M дозволяє передавати файли високої роздільної здатності обсягом до 50 мегабайтів. Запити з префіксом /api/ перенаправляються до Java-оркестратора, а статичні файли інтерфейсу користувача віддаються безпосередньо з локальної файлової системи контейнера. Загальну топологію мережевих зв’язків та схему розгортання інфраструктурних компонентів представлено на рисунку 3.11.

Для тестування системи розроблено імітаційну зовнішню платформу (Mock Service) на мові Python, яка ініціює процес перевірки та приймає результати верифікування через зворотні виклики (webhook). Головний екран імітаційного сервісу містить кнопку запуску процесу та відображає поточний статус ідентифікації. При запуску система формує запит до оркестратора, отримує динамічне посилання та виводить його у модальному вікні (рис. 3.12).

Оскільки аналізування біометричних даних триває кілька секунд, синхронне очікування є неефективним. Зв’язок між серверами реалізовано через асинхронні HTTP-сповіщення (зворотні виклики). Оброблення результатів на стороні тестового клієнта здійснюється методом, який зчитує

JSON-пакет, вилучає статус, причину відмови та розпізнані дані паспорта, після чого оновлює стан у базі даних SQLite (лістинг 3.14).

Конструкція ON CONFLICT DO UPDATE забезпечує ідемпотентність оброблення повідомлень при можливих повторних надсиланнях через мережеві збої. Головний екран оновлює свій стан за допомогою періодичного опитування локального сервера, змінюючи статус на «ВЕРИФІКОВАНО» та виводячи розпізнані текстові дані паспорта (рис. 3.13).

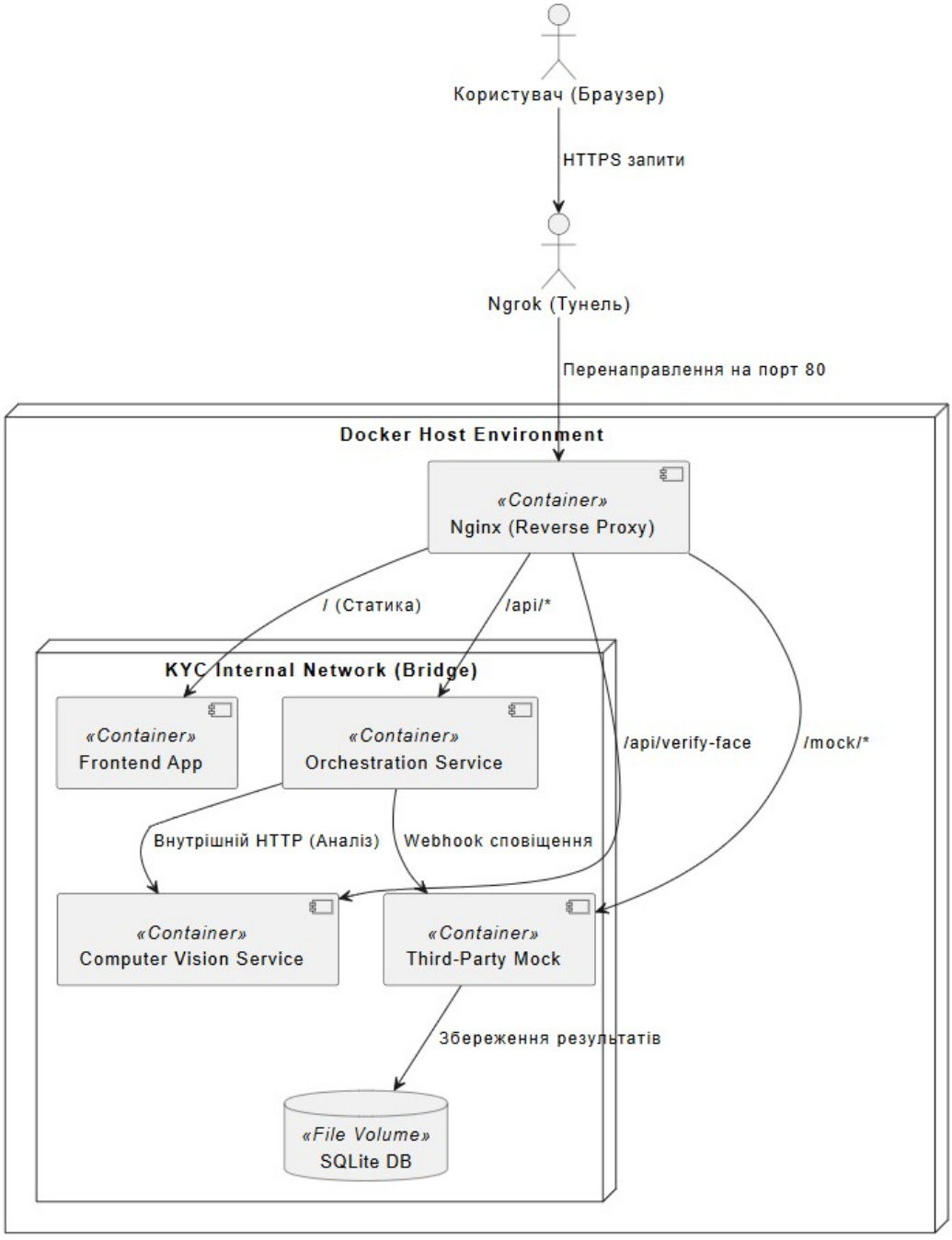

Рисунок 3.11 – Діаграма розгортання інфраструктурних компонентів системи

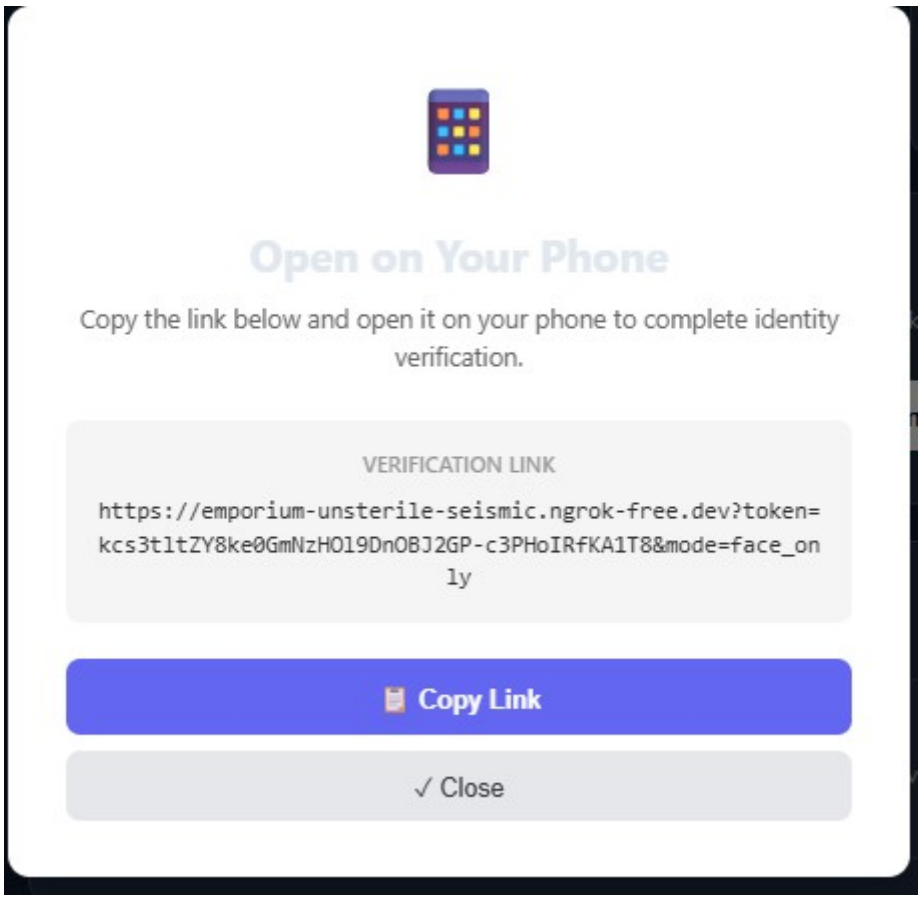

Рисунок 3.12 – Схема інтерфейсу тестової системи з посиланням на ініційовану сесію

Лістинг 3.14 Оброблення вхідного сповіщення (webhook) у тестовій системі:

```
def _api_post_callback_result(self):
    payload = self._read_json()
    status = (str(payload.get(«finalStatus»)).strip().upper() or None)
    fail_reason = payload.get(«failReason»)
    passport = payload.get(«passport»)

    if status is None:
        return self._respond(200, {«status»: «ok», «stored»: False, «reason»: «missing_status»})
```

```
        with get_db() as conn:
            user_id = resolve_user_id_for_callback(conn, payload)
            if user_id is None:
                return self._respond(200, {«status»: «ok», «stored»: False, «reason»: «user_not_found»})

            conn.execute(
                ««««
                INSERT INTO kyc_state(user_id, status, fail_reason, passport_json, updated_at)
                VALUES (?, ?, ?, ?, ?)
                ON CONFLICT(user_id) DO UPDATE SET
                    status = excluded.status,
                    fail_reason = excluded.fail_reason,
                    passport_json = excluded.passport_json,
                    updated_at = excluded.updated_at
                ««««,
                (user_id, status, fail_reason, json.dumps(passport) if passport else None, int(time.time() * 1000)),
            )
        self._respond(200, {«status»: «ok», «stored»: True})
```

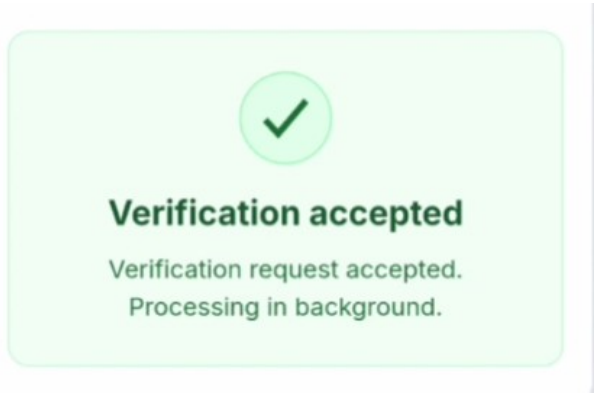

Рисунок 3.13 – Відображення успішних результатів верифікації в інтегрованій системі

3.6 Тестування системи

Для підтвердження працездатності розробленого програмного комплексу проведено комплексне тестування, що охоплювало оцінювання точності алгоритмів машинного навчання, перевіряння інтеграційної взаємодії мікросервісів та вимірювання продуктивності під навантаженням. Експерименти проводилися на репрезентативній вибірці, що містила легітимні біометричні дані, копії документів та матеріали для імітування атак.

Першим етапом було тестування модуля визначання присутності людини (Liveness Detection) на вибірці зі 150 легітимних сесій та 130 спроб шахрайства різних типів. Результати тестування наведено в таблиці 3.4.

Таблиця 3.4 – Результати тестування алгоритмів Liveness Detection

Тип тестового сценарію	**Кількість спроб**	**Успішно розпізнано**	**Помилково розпізнано**	**Точність, %**
Легітимні користувачі	150	142	8 (Хибна відмова)	94,67
Паперова копія (Paper Print)	50	50	0 (Хибний допуск)	100,00
Відтворення з екрана (Screen Replay)	50	49	1 (Хибний допуск)	98,00
Плоска маска (2D Mask)	30	28	2 (Хибний допуск)	93,33

Коефіцієнт хибного допуску (FAR) склав менше 2%, що підтверджує стійкість системи до спроб шахрайства без використання інфрачервоних сенсорів. Коефіцієнт хибної відмови (FRR) становив 5,33%. Основними

причинами помилкових відмов стали недостатнє освітлення та розмиття кадру через різкі рухи головою.

Оцінювання якості алгоритму порівнювання облич (Face Matching) проводилося розрахунком евклідової відстані у 128-вимірному просторі ознак. Прийняті пороги: 0,46 для автоматичного схвалення та 0,55 – для ручного перевіряння. Для легітимних пар середня відстань склала від 0,32 до 0,42. При зміні зовнішності (наприклад, суттєва різниця у віці з фото в паспорті) відстань зростала до 0,48…0,52, спрямовуючи сесію на ручний перегляд. Для різних людей відстань стабільно перевищувала 0,60. Розподіл відстаней показано на рисунку 3.14.

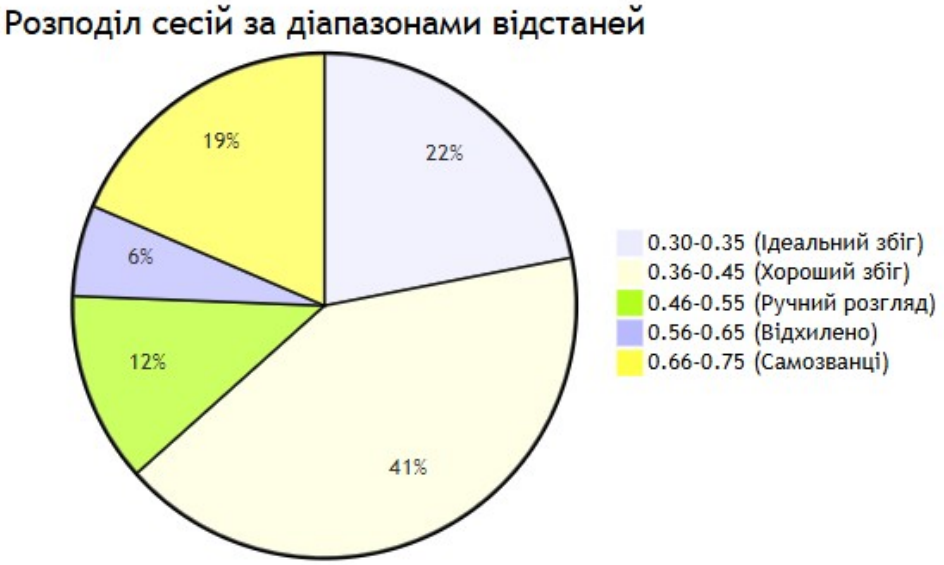

Рисунок 3.14 – Кругова діаграма розподілу евклідових відстаней під час порівняння облич

Для тестування підсистеми оптичного розпізнавання символів (OCR) використано 70 копій документів. Частина з них мала дефекти (відблиски, геометричні викривлення). Точність розпізнавання та валідування наведено в таблиці 3.5.

Таблиця 3.5 – Точність розпізнавання та валідації текстових атрибутів документа

Атрибут документа	**Точність зчитування, %**	**Точність валідування, %**	**Примітки**
Номер документа	97,1	100,0	Стійкість за рахунок контрольного числа
Дата народження	95,5	98,0	Застосовано алгоритм евристичного пошуку
Код країни	98,5	100,0	Чітка стандартизація літер ICAO
Прізвище та ім’я	92,0	–	Контрольні цифри відсутні

При інтеграційному тестуванні вимірювалися часові затримки на кожному етапі оброблення запитів у середовищі Docker на стандартному процесорі (без GPU). Загальний час оброблення на серверній частині становить від 2,5 до 4 секунд. Найбільш ресурсомістким виявився етап нейромережевого аналізування та OCR на базі Python (28% часу). Етап клієнтського знімання та таймерів триває в середньому 6…8 секунд (60% від усього часу взаємодії користувача з системою). Розподіл часових витрат представлено на круговій діаграмі (рис. 3.15).

Логування взаємодії та успішного проходження запитів між основними мікросервісами, сервісом комп’ютерного зору та імітаційною системою зафіксовано на консольній схемі (рис. 3.16).

Усі проведені випробування підтверджують високу надійність та точність інтегрованих алгоритмів. Впровадження комбінованого тестування на базі аналізування контрольних точок забезпечило стійкість до спроб обходу системи без залучення спеціалізованого обладнання.

Рисунок 3.15 – Кругова діаграма розподілу часу виконання процесу верифікації

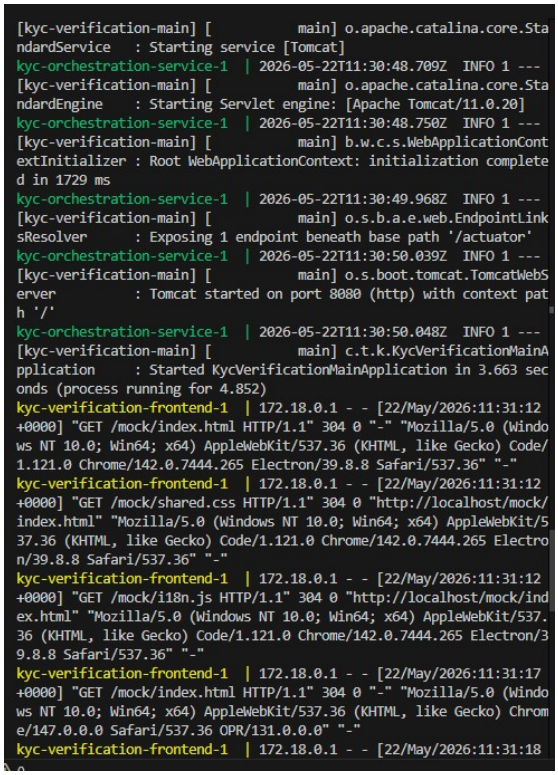

Рисунок 3.16 – Логи виконання HTTP-запитів між мікросервісами під час тестування

Модуль розпізнавання символів за рахунок дублювання аналітичних моделей та автоматичної перевірки контрольних сум продемонстрував високу стійкість до цифрових перешкод та готовий до впровадження у промислову експлуатацію.

ВИСНОВКИ

У рамках кваліфікаційної роботи розроблено і реалізовано програмний комплекс для автоматизованої віддаленої верифікації особи користувача (KYC). Робота охоплює повний цикл створення застосунку від аналізу наявних рішень до реалізації функціоналу із використанням сучасних інструментів, та вирішено такі завдання:

− проведено комплексний аналіз предметної області та процесу ідентифікації клієнтів, що дало можливість дослідити наявні інформаційні технології розпізнавання документів та облич;

− виявлено критичні обмеження закритих комерційних систем верифікації, що дало можливість обґрунтувати доцільність застосування розподіленої мікросервісної інфраструктури для безпечного оброблення конфіденційних мультимедійних даних;

− обрано та теоретично обґрунтовано методи комп’ютерного зору, що дало можливість виконати оптичне розпізнавання символів (OCR) та біометричний аналіз облич;

− спроєктовано архітектуру системи та розмежовано ролі базового керуючого сервісу, підсистеми оброблення зображень і клієнтського вебзастосунку, що дозволило побудувати масштабовану модульну інфраструктуру;

− розроблено алгоритми взаємодії та протоколи обміну даними з використанням архітектурного стилю REST та механізму зворотних викликів, що забезпечило стабільний та надійний інформаційний обмін;

− змодельовано структуру бази даних, що дало можливість забезпечити надійне збереження станів сесій верифікації;

− здійснено вибір інструментальних засобів та програмну реалізацію всіх модулів системи, включаючи основний сервіс управління, ізольований сервіс комп’ютерного зору та інтерфейс кінцевого користувача, що дозволило інтегрувати зазначені компоненти в єдиний комплекс;

– проведено комплексне наскрізне тестування системи та оцінено якість роботи алгоритмів розпізнавання, що дало можливість перевірити працездатність системи в умовах, наближених до реального використання.

Тестування на споживчому обладнанні підтвердило стабільність розпізнавання документів та облич, а також наскрізну взаємодію компонентів. Перспективи розвитку системи полягають в оптимізації швидкодії алгоритмів комп'ютерного зору, розширенні переліку документів, впровадженні додаткових біометричних факторів захисту та розробленні мобільних версій застосунку.

Практична цінність результатів полягає у створенні ресурсоефективного ПЗ для автоматизації віддаленої верифікації (KYC) та безпечного локального оброблення конфіденційних даних без втрати якості розпізнавання.

Результати роботи апробовано у вигляді трьох тез доповідей: під час 29-го Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті» [48]; під час II Міжнародної науково-практичної студентської конференції «ІТ-простір сьогодення: тенденції, інновації та перспективи розвитку» [49]; під час 30-го Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті» [50].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. FATF. (2023). *The FATF Recommendations. International standards on combating money laundering and the financing of terrorism & proliferation* (Updated 2023).

2. Bodyanskiy, Y., Vynokurova, O., Kobylin, I., & Kobylin, O. (2016). Adaptive fuzzy clustering of short time series with unevenly distributed observations in Data Stream Mining tasks. *Information Technology and Management Science*, 23–28.

3. Кобилін, І. О., & Ніколайчук, А. І. (2024). Monitoring and diagnosing faults in online mode using time series data. *Системи обробки інформації*, (3(178)), 27–32.

4. ICAO. (2021). *Doc 9303. Machine readable travel documents* (8th ed.). International Civil Aviation Organization.

5. ISO/IEC. (2017). *ISO/IEC 30107-3:2017. Information technology – Biometric presentation attack detection – Part 3: Testing and reporting*. International Organization for Standardization.

6. Bodyanskiy, Y., Vynokurova, O., Szymański, Z., Kobylin, I., & Kobylin, O. (2016, August). Adaptive robust models for identification of nonstationary systems in data stream mining tasks. In *2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP)* (pp. 263–268). IEEE.

7. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778).

8. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. *Advanced Information Systems*, *9*(1), 5–12.

9. Daradkeh, Y.I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2024). Improving the effectiveness of image classification structural methods by

compressing the description according to the information content criterion. *Computers, Materials & Continua*, *80*(2), 3085–3106.

10. Gorokhovatskyi, V., & Tvoroshenko, I. (2023). Identification of visual objects by the search request. In *International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium, September 28, 2023, Kyiv-Uzhorod, Ukraine* (pp. 25–27).

11. Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., & Hudáková, M. (2025). Image description compression in classification structural methods. *IEEE Access*, *13*, 43631–43641.

12. Боденчук-Пастухов, Є. В., & Кобилін, І. О. (2026). Оцінка моделей трансформерів машинного перекладу на низько-ресурсних, азійсько-українських мовних парах. *Системи обробки інформації*, (1(184)), 116–123.

13. Gorokhovatskyi, V., Tvoroshenko, I., & Yakovleva, O. (2024). Transforming image descriptions as a set of descriptors to construct classification features. *Indonesian Journal of Electrical Engineering and Computer Science*, *33*(1), 113–125.

14. Smith, R. (2007). An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)* (pp. 629–633).

15. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Handwritten character recognition models based on convolutional neural networks. *International Journal of Academic Engineering Research*, *7*(9), 64–72.

16. Творошенко, І. С. (2025). Оцінювання значущості ознак для підвищення продуктивності структурних методів класифікації зображень. У *Проблеми інформатики та моделювання (ПІМ-2025). Тези двадцять п'ятої міжнародної науково-технічної конференції (25–28 вересня 2025 року)* (с. 38–43). НТУ «ХПІ».

17. Tvoroshenko, I., Gorokhovatskyi, V., Kobylin, O., & Tvoroshenko, A. (2023). Application of deep learning methods for recognizing and classifying culinary dishes in images. *International Journal of Academic and Applied Research*, *7*(9), 57–70.

18. Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., & Hudáková, M. (2026). Feature space quantization and application of metrics in structural methods of image recognition. *IEEE Access*, *14*, 17812–17824.

19. Кобилін, І. О., & Харченко, А. І. (2024). Classification techniques for computer vision. *Системи обробки інформації*, (3(178)), 33–41.

20. Kharchenko, A. I., & Kobylin, I. O. (2024). *Performance analysis of Support Vector Machine for vehicle classification*. V. N. Karazin Kharkiv National University.

21. Daradkeh, Y.I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Tools for fast metric data search in structural methods for image classification. *IEEE Access*, *10*, 124738–124746.

22. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks. *International Journal of Academic Information Systems Research*, *7*(7), 25–36.

23. Tvoroshenko, I., Pomazan, V., Gorokhovatskyi, V., & Kobylin, O. (2023). Application of video data classification models using convolutional neural networks. *International Journal of Academic and Applied Research*, *7*(11), 134–145.

24. Matarneh, R., Tvoroshenko, I., & Lyashenko, V. (2019). Improving fuzzy network models for the analysis of dynamic interacting processes in the state space. *International Journal of Recent Technology and Engineering*, *8*(4), 1687–1693.

25. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description. *Advances in Electrical and Electronic Engineering*, *21*(1), 19–27.

26. Larin, I., Tvoroshenko, I., Gorokhovatskyi, V., & Liubchenko, V. (2025). Research and comparison of facial color normalization methods relative to an etalon image. *International Journal of Academic and Applied Research*, *9*(11), 36–47.

27. Tvoroshenko, I. (2019). *Development of models of spatial analysis of status of interactive processes of complex systems*.

28. Karakonstantyn, D., & Tvoroshenko, I. (2024). About the issue of optimization the performance of the server part of the information system. In *Abstracts of XII International Scientific and Practical Conference* (pp. 364–366).

29. Кучеренко, Є. І., & Творошенко, І. С. (2011). Оперативне оцінювання простору станів складних розподілених об’єктів з використанням нечіткої інтервальної логіки. *Искусственный интеллект*, (3), 382–387.

30. Кучеренко, Е. И., Филатов, В. А., Творошенко, І. С., & Байдан, Р. Н. (2005). Интеллектуальные технологии в задачах принятия решений технологических complexes на основе нечеткой интервальной логики. *Восточно-Европейский журнал передовых технологий*, (2), 92–96.

31. Гороховатський, В., & Творошенко, І. (2026). *Продуктивні моделі аналізу даних у методах розпізнавання зображень: монографія*. ХНУРЕ.

32. Kobylin, O., Gorokhovatskyi, V., Tvoroshenko, I., & Peredrii, O. (2020). The application of non-parametric statistics methods in image classifiers based on structural description components. *Telecommunications and Radio Engineering*, *79*(10), 855–863.

33. Бодянський, Є., Винокурова, О., Кобилін, І., & Мулеса, П. (2017). Адаптивна матрична нейро-фаззі самоорганізовна мережа для кластеризації багатовимірних потоків даних. *Вісник Національного університету «Львівська політехніка»*, (864), 314–319.

34. Гороховатський, В., Передрій, О., Творошенко, І., & Марков, Т. (2023). Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень. *Сучасні інформаційні системи*, *7*(1), 5–13.

35. Гороховатський, В. О., & Творошенко, І. С. (2022). *Аналіз багатовимірних даних за описом у формі множини компонент: монографія*. ХНУРЕ.

36. Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., Hudáková, M., & Gorokhovatskyi, O. (2024). Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set. *IEEE Access*, *12*, 73376–73385.

37. Daradkeh, Y.I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., & Al-Dhaifallah, M. (2023). Statistical data analysis models for determining the relevance of structural image descriptions. *IEEE Access*, *11*, 126938–126949.

38. Tvoroshenko, I. S., & Gorokhovatsky, V. O. (2019). Intelligent classification of biophysical system states using fuzzy interval logic. *Telecommunications and Radio Engineering*, *78*(14), 1303–1315.

39. Lyashenko, V., Mustafa, S. K., Tvoroshenko, I., & Ahmad, M. A. (2020). Methods of using fuzzy interval logic during processing of space states of complex biophysical objects. *International Journal of Emerging Trends in Engineering Research*, *8*(2), 372–377.

40. Yakovleva, O., Matúšová, S., Tvoroshenko, I., & Isaiev, Y. (2024). Visitor counting based on video stream analysis from surveillance cameras to solve various business problems. *Verejná správa a regionálny rozvoj – ekonómia, manažment a marketing*, *XX*(1), 67–87.

41. Гороховатський, В. О., Творошенко, І. С., & Чмутов, Ю. В. (2022). Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень. *Сучасні інформаційні системи*, *6*(3), 5–12.

42. Бодянський, Є. В., Винокурова, О. А., Ізонін, І. В., Кобилін, І. О., & Мулеса, П. П. (2017). *Кластеризація багатовимірних часових рядів на основі адаптивної матричної нейро-фаззі самоорганізовної мережі*. Intellectual Systems For Decision Making.

43. Suprun, A., Tvoroshenko, I., Gorokhovatskyi, V., & Yakovleva, O. (2025). Development and research of a method for the combined use of large language models for text generation. *International Journal of Academic and Applied Research*, *9*(10), 249–263.

44. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions on a variety of effective factors using fuzzy logic. *International Journal of Advanced Computer Science and Applications Russo*, *11*(5), 43–50.

45. Творошенко, І. С. (2021). *Технології прийняття рішень в інформаційних системах: навч. посібник*. ХНУРЕ.

46. Творошенко, И. С. (2010). Анализ процессов принятия решений в интеллектуальных системах. *Системи обробки інформації*, (2), 248–253.

47. Bohdan, N., Tvoroshenko, I., Gorokhovatskyi, V., & Kobylin, O. (2025). Development of a hybrid method to enhance context memory for a chatbot application based on large language models. *International Journal of Academic Information Systems Research*, *9*(10), 7–18.

48. Матвєєв, Є. Є. (2025). Переваги і недоліки роботи з різними версіями Java. У *29-й Міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI столітті»: Зб. матеріалів форуму*. ХНУРЕ.

49. Матвєєв, Є. Є. (2025). Використання технології Face ID для автентифікації користувачів. У *IT-простір сьогодення: тенденції, інновації та перспективи розвитку: збірник тез доповідей II МНПК*. ХНУ імені В. Н. Каразіна.

50. Матвєєв, Є. Є. (2026). Принципи функціонування та переваги технології Face ID. У *30-й Міжнародний молодіжний форум «Радіоелектроніка і молодь у XXI столітті»: Зб. матеріалів форуму*. ХНУРЕ.