

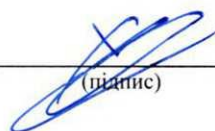
**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: Алгоритмічне забезпечення системи оптичного розпізнавання символів (OCR) для телеметрії в месенджерах.

Здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи КБ-22-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за галузями)»


(підпис)

Харченка Сергія
Олександровича
(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Горбовий Артур Юліанович,
д.т.н, професор

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

Керівник освітньо-
професійної програми


(підпис)

Ніжегородцев Владислав
Олександрович, канд. пед.
наук, доцент

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

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних технологій і систем протокол від «12» травня 2026 № 20

Завідувач кафедри


(підпис)

Філоненко Михайло
Миколайович, канд. фіз.-мат.
наук, доцент

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

Ірпінь – 2026

Реєстрація № 61

28.05.2026
(дата)


(підпис відповідальної особи)

Результати перевірки

до захисту
(до захисту, на доопрацювання)

12.05.2026
(дата)


(підпис)

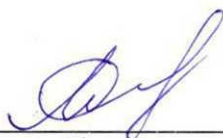
Результат захисту

3 (задовільно)
(оцінка)

08.06.2026
(дата)

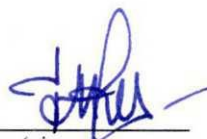

(підпис)

Голова комісії:


(підпис)

О.С. Дюбанов

Члени комісії:


(підпис)

М.М. Філоненко


(підпис)

В.О. Ніжегородцев


(підпис)

А.А. Омельчук

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	18		08.06.2016 Горбовий А.Ю. Дюбанов О.С.
2. Дотримання вимог щодо оформлення	від 0 до 10	7		Філоненко М.М. Ніжегородцев В.О.
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		Омельчук А.А. Суховерхівка Л.П.
4. Рецензент	від 0 до 5	3		
РАЗОМ БАЛІВ	від 0 до 50	32		

Оцінювання кваліфікаційної роботи науковим керівником – 33 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підписи)
65	D	3 (задовільно)	08.06.2016	Дюбанов О.С. Філоненко М.М. Ніжегородцев В.О. Омельчук А.А. Суховерхівка Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	3 (задовільно)
50-59	E	
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій

Кафедра комп'ютерних та інформаційних технологій і систем

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 122 «Комп'ютерні науки»

Освітня програма: «Інформаційні управляючі системи і технології (за галузями)»

Затверджую

Завідувач кафедри Філоненко М.М.

«01» грудня 2025



Завдання на кваліфікаційну роботу

здобувачу вищої освіти

Харченка Сергія Олександровича

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи Алгоритмічне забезпечення системи оптичного розпізнавання символів (OCR) для телеметрії в месенджерах
Затверджена наказом ректора від «28» листопада 2025 №1526

2. Строк здачі закінченої роботи «08» травня 2026

3. Мета роботи створення Android-застосунку для OCR-розпізнавання показників тонометра Omron, їх локального збереження, статистичного аналізу та генерації AI-висновків.

Об'єкт дослідження цифрова фіксація, збереження та аналіз телеметричних показників артеріального тиску.

Предмет дослідження алгоритми і програмні засоби розпізнавання, збереження та аналізу телеметрії в Android-середовищі.

Орієнтована база дослідження розроблений застосунок «OmronOcrScanner».

4. Вихідні дані до роботи мова Kotlin, CameraX, Room DB, Gemini API, метрики тиску і пульсу (SYS, DIA, Pulse).

5. Перелік графічного матеріалу архітектура MVVM, структура бази даних, конвеєр OCR-обробки, скріншоти інтерфейсу (камера, аналітика, історія, налаштування).

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)
ВСТУП

РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ СИСТЕМ ОПТИЧНОГО РОЗПІЗНАВАННЯ СИМВОЛІВ У ТЕЛЕМЕТРІЇ
1.1. Аналіз проблеми цифровізації медичних даних та сучасних стандартів моніторингу. 1.2. Архітектурні особливості та математичний апарат OCR для семисегментних РК-дисплеїв. 1.3. Порівняльний аналіз локальних та хмарних технологій комп'ютерного зору.

РОЗДІЛ 2. ПРОЄКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ЗЧИТУВАННЯ ТА АНАЛІЗУ ПОКАЗНИКІВ ТОНОМЕТРА OMRON
2.1. Функціональні вимоги та сценарії використання застосунку. 2.2. Архітектура програмного забезпечення та структура даних. 2.3. Алгоритми розпізнавання, валідації та статистичної обробки вимірювань.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ПЕРЕВІРКА ANDROID-ЗАСТОСУНКУ ДЛЯ ОБРОБКИ ТЕЛЕМЕТРИЧНИХ ПОКАЗНИКІВ
3.1. Реалізація модулів отримання, розпізнавання та збереження даних.
3.2. Реалізація модулів статистики, історії вимірювань і AI-висновків.
3.3. Тестування роботи застосунку та оцінювання практичних результатів.
ВИСНОВКИ. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ
ДОДАТКИ

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Строк виконання етапів роботи
		За планом	Фактично
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 30.10.2025	До 30.10.2025
2	I розділ (Теоретико-методологічні основи: аналіз проблеми цифровізації медичних даних, архітектурні особливості OCR для РК-дисплеїв та порівняння технологій комп'ютерного зору)	26.12.2025	26.12.2025
3	II розділ (Проектування мобільного застосунку: формування вимог, розробка архітектури MVVM, структури бази даних Room та алгоритмів розпізнавання і валідації)	18.02.2026	18.02.2026
4	III розділ (Реалізація та тестування: програмування Android-застосунку, інтеграція CameraX та Gemini API, реалізація аналітики та проведення натурних експериментів)	16.04.2026	16.04.2026
5	Формування загальних висновків, оформлення додатків та списку використаних джерел	01.05.2026	01.05.2026
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	07.05.2026	07.05.2026
7	Отримання відгуку керівника	29.05.2026	29.05.2026
8	Отримання зовнішньої рецензії	01.06.2026	01.06.2026

8. Дата видачі завдання «01» грудня 2025

9. Керівник кваліфікаційної роботи


(підпис)

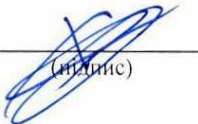
Горбовий А.Ю.
(прізвище, ініціали)

10. Керівник освітньо-професійної програми


(підпис)

Ніжегородцев В.О.
(прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти


(підпис)

Харченко С.О.
(прізвище, ініціали)

АНОТАЦІЯ

Алгоритмічне забезпечення системи оптичного розпізнавання символів (OCR) для телеметрії

Мета роботи полягає в розробці Android-застосунку для автоматизованого розпізнавання показників тонометра Omron, локального збереження даних, статистичного аналізу та генерації AI-висновків.

Завданнями роботи є аналіз предметної області; обґрунтування архітектури OCR; проєктування застосунку; розробка каскадного алгоритму з CameraX та Gemini API; інтеграція бази Room; проведення тестування.

Предметом дослідження є алгоритми та програмні засоби розпізнавання, збереження й аналізу телеметрії в Android-застосунку.

Об'єктом дослідження виступає процес цифрової фіксації та аналізу телеметричних показників із медичних приладів.

Базою дослідження виступає мобільний застосунок OmronOcrScanner.

За результатами дослідження створено архітектуру та код застосунку, що забезпечує каскадне OCR-розпізнавання екранів, збереження історії та генерацію аналітичних звітів.

Одержані результати можуть бути використані для зручного моніторингу здоров'я та подальшої розробки mHealth-рішень.

Кваліфікаційна робота містить 60 сторінок, 1 таблицю, 4 рисунка, 40 джерел. Ключові слова OCR, Android, Kotlin, комп'ютерний зір, бази даних, mHealth, Gemini API.

Рік виконання кваліфікаційної роботи 2026

Рік захисту роботи 2026

ABSTRACT

Algorithmic support of the optical character recognition (OCR) system for telemetry

The aim of the work is the development of an Android application for automated recognition of Omron blood pressure monitor indicators, local data storage, statistical analysis, and the generation of AI insights.

The tasks of the work are analysis of the subject area; justification of the OCR architecture; application design; development of a cascading algorithm using CameraX and Gemini API; integration of the Room database; conducting testing.

The subject of the research is the algorithms and software tools for the recognition, storage, and analysis of telemetry in an Android application.

The object of the research is the process of digital recording and analysis of telemetry indicators from medical devices.

The research base is the OmronOcrScanner mobile application.

The results of the research include the architecture and code of the application were created, providing cascading OCR screen recognition, history storage, and the generation of analytical reports.

The obtained results can be used for convenient health monitoring and further development of mHealth solutions.

The qualification work consists of 60 pages, 1 table, 4 figures, and 40 references.

Keywords: OCR, Android, Kotlin, computer vision, databases, mHealth, Gemini API.

Year of qualification work : 2026

Year of work defense: 2026

ЗМІСТ

ВСТУП.....	9
РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ СИСТЕМ ОПТИЧНОГО РОЗПІЗНАВАННЯ СИМВОЛІВ У ТЕЛЕМЕТРІЇ.....	12
1.1. Аналіз проблеми цифровізації медичних даних та сучасних стандартів моніторингу	12
1.2. Архітектурні особливості та математичний апарат OCR для семисегментних РК-дисплеїв.....	15
1.3. Порівняльний аналіз локальних та хмарних технологій комп'ютерного зору	17
РОЗДІЛ 2. ПРОЄКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ЗЧИТУВАННЯ ТА АНАЛІЗУ ПОКАЗНИКІВ ТОНОМЕТРА OMRON	23
2.1. Функціональні вимоги та сценарії використання застосунку	23
2.2. Архітектура програмного забезпечення та структура даних	30
2.3. Алгоритми розпізнавання та статистичної обробки вимірювань	32
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ПЕРЕВІРКА ANDROID-ЗАСТОСУНКУ ДЛЯ ОБРОБКИ ТЕЛЕМЕТРИЧНИХ ПОКАЗНИКІВ	36
3.1. Реалізація модулів отримання, розпізнавання та збереження даних.....	36
3.2. Реалізація модулів статистики, історії вимірювань і AI-висновків	39
3.3. Тестування роботи застосунку та оцінювання практичних результатів ..	41
ВИСНОВКИ	45
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	48
ДОДАТКИ.....	52

ВСТУП

Сучасний світ характеризується стрімким впровадженням цифрових технологій у всі сфери життєдіяльності людини. Особливий вплив мобільні інновації мають на галузь охорони здоров'я та персонального медичного моніторингу (mHealth). Пацієнти дедалі частіше використовують смартфони для відстеження показників свого організму, контролю хронічних станів та профілактики захворювань. Систематичний контроль артеріального тиску є критично важливим для підтримки здоров'я серцево-судинної системи, тому інструменти, що здатні автоматизувати збір телеметричних медичних даних безпосередньо в домашніх умовах, набувають сьогодні надзвичайної актуальності.

Актуальність теми. В умовах інтенсивного темпу життя людям буває важко вести регулярний паперовий щоденник артеріального тиску та пульсу. Більшість надійних побутових тонометрів обладнані монохромними рідкокристалічними екранами та не мають вбудованих засобів для бездротової передачі даних. Користувачі змушені вручну переписувати показники, що часто призводить до механічних помилок, плутанини або втрати інформації. Тому створення сучасного мобільного застосунку, який використовує технології оптичного розпізнавання символів (OCR) для миттєвого зчитування даних з екрана, надійно зберігає історію локально та формує зрозумілу статистику, є актуальним і своєчасним завданням. Такий інструмент здатен значно спростити процес щоденного моніторингу та підвищити прихильність пацієнтів до контролю власного здоров'я.

Стан і ступінь розробки проблеми. У науковій літературі та на ринку програмного забезпечення широко представлені праці, присвячені системам комп'ютерного зору. Відзначаються вагомі розробки у сфері сканування друкованого тексту та загальних OCR-бібліотек. Проте питання стабільного розпізнавання специфічних розривних семисегментних індикаторів на екранах медичних приладів (особливо за умов відблисків та слабкого контрасту) все ще

потребує вдосконалення. Створення надійного конвеєра, що поєднує локальне кадрування відеопотоку та контекстний аналіз за допомогою сучасних хмарних мультимодальних моделей штучного інтелекту, залишається відкритим і потребує адаптації під потреби користувачів мобільних систем.

Метою кваліфікаційної роботи є створення мобільного застосунку на ОС Android для автоматизованого ведення щоденника артеріального тиску, що забезпечить покращення користувацького досвіду шляхом швидкого оптичного розпізнавання показників тонометра та оптимізації статистичного аналізу даних.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- Проаналізувати предметну область, дослідити існуючі алгоритми комп'ютерного зору та визначити ключові вимоги до мобільної OCR-системи;
- Обґрунтувати вибір платформи, мови програмування та архітектури бази даних;
- Створити інтуїтивно зрозумілий та сучасний користувацький інтерфейс для зручної взаємодії під час сканування;
- Реалізувати локальну обробку відеопотоку та просторове кадрування зони інтересу за допомогою інструментів CameraX;
- Налаштувати взаємодію з хмарним сервісом Gemini API для високоточного розпізнавання показників з монохромних дисплеїв;
- Впровадити локальну систему збереження та агрегації телеметричних даних на базі СУБД Room для роботи в офлайн-режимі;
- Провести тестування функціоналу застосунку та оцінити точність розпізнавання в реальних умовах експлуатації.

Об'єктом кваліфікаційної роботи є процес проєктування та створення систем автоматизованого збору телеметричних медичних даних.

Предмет дослідження - архітектура, алгоритми розпізнавання та програмна реалізація мобільного Android-застосунку для автоматизації збору телеметрії.

Методи дослідження - теоретичні - аналіз науково-технічної літератури, узагальнення інформації щодо стандартів mHealth та методів комп'ютерного зору, порівняння існуючих OCR-рішень; *емпіричні* - проєктування інтерфейсу користувача, об'єктно-орієнтоване програмування, цифрова обробка зображень, тестування програмного забезпечення та налагодження взаємодії з API.

Базою дослідження виступає створений мобільний застосунок OmronOcrScanner.

Науково-практична значимість одержаних результатів. Створений застосунок вирішує практичну проблему автоматизації обліку життєво важливих показників організму. Використання гібридного каскадного алгоритму розпізнавання (просторове кадрування + ШІ) дозволяє успішно нівелювати оптичні завади екранів тонометрів. Інтеграція надійної локальної бази даних гарантує конфіденційне зберігання історії вимірювань та швидку роботу без постійного доступу до інтернету, а статистична панель дозволяє швидко оцінити динаміку тиску. Одержані результати можуть бути використані користувачами для щоденного контролю здоров'я, а застосовані архітектурні підходи - IT-спеціалістами у подальшому створенні подібних медичних мобільних сервісів.

Особистий внесок здобувача полягає у повному циклі створення мобільного застосунку OmronOcrScanner: від розробки концепції та архітектури (MVVM) до написання програмного коду мовою Kotlin. Автором самостійно спроектовано локальну реляційну базу даних на основі Room, реалізовано алгоритми попередньої нормалізації відеокадрів (з використанням CameraX), налаштовано мережеву інтеграцію з моделлю Gemini API для OCR-обробки та створено модуль аналітичної агрегації телеметричних показників.

РОЗДІЛ 1. ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ СИСТЕМ ОПТИЧНОГО РОЗПІЗНАВАННЯ СИМВОЛІВ У ТЕЛЕМЕТРІЇ

1.1. Аналіз проблеми цифровізації медичних даних та сучасних стандартів моніторингу

Процес масштабної цифровізації сфери охорони здоров'я на сучасному етапі розвитку інформаційного суспільства вимагає побудови надійних, безпечних та безперервних каналів передачі інформації від первинного джерела, яким виступає пацієнт, до централізованої медичної інформаційної системи. Глобальні ініціативи Всесвітньої організації охорони здоров'я визначають постійний контроль за серцево-судинними захворюваннями як один із найвищих пріоритетів для збереження тривалості та якості життя населення, особливо в умовах старіння нації [21]. У контексті медичної телеметрії, яка передбачає дистанційне вимірювання та збір ключових фізичних показників організму, найважливішим параметром є підтримка абсолютної актуальності даних у часі та мінімізація будь-яких затримок між моментом безпосереднього вимірювання і моментом потрапляння цих даних до електронної карти пацієнта для подальшого аналізу лікарем. Європейське товариство кардіологів у своїх наукових документах щодо профілактики серцево-судинних захворювань наголошує на критичній необхідності регулярного самостійного моніторингу артеріального тиску в домашніх умовах [20]. Проте сучасний ринок побутової медичної техніки залишається значною мірою консервативним. Він заповнений десятками мільйонів одиниць високоточних, надійних, але технологічно ізольованих приладів, таких як тонометри, глюкометри та електронні термометри. Наприклад, класичні пристрої провідних виробників, експлуатаційна документація яких детально регламентує процеси вимірювання, здебільшого розраховані на виключно візуальне відображення результатів без можливості їх автоматичного експорту [8]. Ці пристрої чудово виконують свою

базову медичну функцію діагностики, однак здебільшого вони не підтримують сучасні загальноприйняті стандарти бездротової передачі медичних даних.

Дослідження багатьох науковців переконливо вказують на те, що архаїчне ручне введення даних залишається головним технічним та психологічним бар'єром для побудови якісних інтелектуальних систем підтримки прийняття лікарських рішень у сфері електронної медицини. Фахівці з інформаційних технологій у фізичній та реабілітаційній медицині зазначають, що без автоматизації збору інформації ефективність будь-якої терапії значно знижується [2]. Пацієнти, які обтяжені постійною необхідністю вести паперові щоденники самоконтролю або вручну переписувати цифри у цифрові форми, стикаються з проблемою втрати мотивації. Як наслідок, вони часто пропускають графік вимірювань, свідомо чи несвідомо округлюють показники до десятків, або вносять абсолютно викривлені дані через втому, порушення зору чи звичайну неуважність. Враховуючи, що основною цільовою аудиторією таких медичних приладів як домашні тонометри є переважно люди похилого віку, проблема механічних помилок під час ручного транскрибування стає не просто технічною незручністю, а фактором, що може призвести до призначення хибного дозування гіпотензивних препаратів. Американська асоціація серця у своїх рекомендаціях щодо розуміння показників артеріального тиску підкреслює, що лише точні щоденні дані можуть слугувати надійною базою для коригування терапевтичних протоколів [40]. Загальний розвиток мобільних технологій для моніторингу показників стану здоров'я пропонує різноманітні рішення для обходу цієї проблеми, проте більшість із них вимагає від пацієнта придбання нових смарт-пристроїв із вбудованими модулями синхронізації [4].

Тому в галузі програмної інженерії виникає нагальна потреба у створенні універсального проміжного програмного шару на базі звичайних персональних смартфонів. Такий підхід повністю відповідає сучасним концепціям інженерії програмного забезпечення, які орієнтовані на максимізацію повторного використання існуючих апаратних ресурсів користувача [37]. Мобільний

застосунок у цьому контексті виконуватиме функцію бездротового інтелектуального шлюзу між ізольованим аналоговим або офлайнним медичним приладом та глобальною електронною системою охорони здоров'я. У такій архітектурній парадигмі смартфон виступає потужним інструментом неінвазивного зчитування інформації: за допомогою вбудованої камери високої роздільної здатності та складних алгоритмів комп'ютерного зору він перетворює візуальне відображення з екрана приладу на жорстко структурований цифровий об'єкт. Цей підхід дозволяє безболісно та економно інтегрувати величезний вже наявний у населення парк медичної техніки в сучасні системи мобільного здоров'я без жодної необхідності змушувати пацієнтів витратити кошти на дорогі прилади нового покоління. Розробка такого програмного забезпечення також вимагає дотримання суворих ергономічних стандартів взаємодії людини та комп'ютерної системи, де ключову роль відіграють принципи юзабіліті та зручності користувацьких інтерфейсів [22]. Адже якщо процес оптичного сканування екрана тонометра виявиться для літньої людини складнішим за використання звичайної кулькової ручки та паперу, впровадження інновації не матиме жодного практичного сенсу [31]. Саме тому фокус розробки зміщується в бік максимальної автоматизації: користувачеві достатньо лише навести камеру на прилад, а всі складні процеси вирівнювання, фільтрації, розпізнавання та перевірки на валідність мають відбуватися у фоновому режимі, непомітно для пацієнта [29].

Окремим і надзвичайно важливим вектором вирішення цієї проблеми є концепція «Thin Client» (тонкого клієнта) на базі популярних месенджерів. Статистика використання смартфонів свідчить, що більшість людей похилого віку успішно користуються програмами для обміну повідомленнями (Telegram, Viber), але зазнають труднощів зі встановленням, налаштуванням та використанням вузькоспеціалізованих медичних додатків. Тому ідеальна архітектура системи збору телеметрії повинна передбачати можливість функціонування у форматі чат-бота. Пацієнту достатньо лише сфотографувати

екран тонометра та відправити це фото у спеціальний чат у месенджері. Далі алгоритмічне забезпечення (серверний OCR-модуль) автоматично обробить зображення, розпізнає цифри та надішле у відповідь текстове повідомлення зі збереженими показниками та медичними рекомендаціями. Таким чином, розробка надійного алгоритмічного ядра, здатного працювати як у нативному Android-додатку, так і виступати бекендом для інтеграції з месенджерами через спеціалізовані API, є найбільш гнучким та універсальним інженерним рішенням.

1.2. Архітектурні особливості та математичний апарат OCR для семисегментних РК-дисплеїв

Проектування алгоритмічного забезпечення для зчитування показників з медичних приладів зіштовхується зі значними викликами, пов'язаними з фізичною природою їхніх дисплеїв. Класичні системи оптичного розпізнавання тексту, які традиційно використовуються для оцифрування друкованих книг, автоматичного сканування офісних документів або розпізнавання автомобільних номерних знаків у транспортному потоці, орієнтовані на обробку типографських шрифтів із безперервними, плавними лініями глифів [39]. Натомість рідкокристалічні дисплеї побутових тонометрів використовують принципово інший, глибоко оптимізований для економії електроенергії підхід до формування цифрового зображення - семисегментну матрицю. Кожна окрема цифра на такому монохромному екрані формується комбінацією лише семи окремих геометричних макроелементів, які розташовані у вигляді стилізованої прямокутної цифри вісім. Дослідники у сфері комп'ютерного зору описують такий символ як специфічний вектор станів сегментів, де кожен сегмент може приймати лише два логічні стани: бути активним (темним) або неактивним (прозорим) [17]. Залежно від конкретної комбінації включених сегментів формуються всі можливі десяткові цифри від нуля до дев'яти, а також деякі службові символи, такі як індикатори серцебиття чи повідомлення про помилку накачування манжети.

Під час обробки таких зображень за допомогою стандартних методів контурного аналізу та машинного розпізнавання виникає низка специфічних математичних, оптичних та алгоритмічних проблем, що унеможливають пряме застосування готових відкритих бібліотек. Першою і найголовнішою перешкодою є топологічна незв'язність, або розриви між сегментами. Звичайні алгоритми виділення ознак та контурного аналізу сприймають одну велику цифру на екрані тонометра не як цілісний зв'язний об'єкт, а як хаотичний набір з кількох окремих розрізнених плям. Це повністю руйнує логіку класичних конволюційних нейромереж розпізнавання, які історично навчалися на суцільних рукописних або друкованих символах [30]. Другою критичною проблемою є ефект так званої дисплейної пам'яті та загальний низький рівень контрасту рідких кристалів. Оскільки матриця тонометра не має власного активного підсвічування і працює на відбиття навколишнього світла від дзеркальної підкладки, неактивні сегменти часто залишають слабкий, але помітний для камери тінювий слід на сірому фоні. При застосуванні стандартної глобальної бінаризації зображення ці слабкі тіні помилково перетворюються на чорні пікселі і розпізнаються комп'ютером як активні елементи [32]. Третім деструктивним фактором виступають постійні перспективні та афінні спотворення. У реальних побутових умовах експлуатації користувач майже ніколи не тримає камеру смартфона строго паралельно площині екрана медичного приладу. Через це виникає геометричне зміщення осей, з'являються сильні оптичні відблиски від кімнатних ламп освітлення на опуклому захисному полікарбонатному склі екрана, що призводить до часткового або повного перекриття корисної інформації [9].

Для ефективного усунення цих завад і підготовки зображення до машинного читання математичний апарат обробки повинен включати багатоетапні кроки попередньої просторової трансформації та складної адаптивної фільтрації. Науковці у сфері розробки систем інтелектуального розпізнавання наголошують, що успіх операції на вісімдесят відсотків залежить

саме від якості алгоритмів передобробки [3]. Тому базові інженерні підходи вимагають використання локально-адаптивних методів порогової бінаризації зображень замість застарілих глобальних порогів. Локальні алгоритми вираховують динамічний поріг яскравості індивідуально для кожного окремого пікселя зображення на основі глибокого статистичного аналізу локального вікна його околиці. Такий підхід дозволяє системі ефективно нівелювати загальну нерівномірність освітлення кімнати, ігнорувати градієнтні тіні та успішно відсікати сліди від неактивних сегментів рідкокристалічної матриці. Після етапу бінаризації зазвичай застосовуються математичні морфологічні операції дилатації (розширення) та ерозії (звуження), які дозволяють штучно об'єднати розірвані сегменти однієї цифри у єдиний зв'язний компонент, придатний для подальшої класифікації традиційними методами комп'ютерного зору [14].

1.3. Порівняльний аналіз локальних та хмарних технологій комп'ютерного зору

Під час архітектурного проєктування будь-якої сучасної мобільної OCR-системи розробник постає перед важливим стратегічним вибором між двома фундаментальними парадигмами виконання обчислень: проведенням усього математичного аналізу та класифікації безпосередньо на процесорі мобільного пристрою (концепція граничних обчислень, або Edge Computing) чи делегуванням цих надзвичайно ресурсоємних задач на потужні віддалені хмарні сервери (Cloud-Based Architecture). Вибір між цими підходами формує основу програмної архітектури застосунку та безпосередньо впливає на його загальну продуктивність, масштабованість, рівень споживання заряду батареї смартфона та, що найважливіше для медичної сфери, на точність отриманих результатів [13]. Кожен з цих підходів має свої беззаперечні переваги та критичні обмеження в специфічному контексті обробки зображень низькоконтрастних медичних телеметричних індикаторів.

Для максимально об'єктивного вибору технологічного стека та створення архітектури, яка б відповідала найвищим стандартам якості програмного забезпечення [23], було проведено ретельний технічний аналіз трьох найбільш популярних рішень на ринку інформаційних технологій. До аналізу увійшли: повністю відкрита локальна бібліотека Tesseract OCR, пропрієтарний оптимізований локальний інструмент розробника Google ML Kit, а також хмарний сервіс нового покоління на базі мультимодальних великих мовних моделей штучного інтелекту Gemini API [15].

Результати комплексного порівняння цих технологій за ключовими критеріями, що є специфічними для розпізнавання телеметричних індикаторів із медичних екранів, зведено у таблицю 1.1.

Таблиця 1.1.

Порівняльна характеристика інструментальних засобів розпізнавання телеметричної інформації з рідкокристалічних дисплеїв

Критерій порівняння	Локальна бібліотека Tesseract OCR	Фреймворк ML Kit (Google Edge Computing)	Мультимодальна хмарна модель Gemini API
1	2	3	4
Точність розпізнавання розривних семисегментних символів	Низька (алгоритми контурного аналізу втрачають об'єкти через незв'язність сегментів)	Середня (висока ймовірність алгоритмічних колізій між цифрами 1 та 7, 0 та 8)	Дуже висока (модель здатна глибоко розуміти семантичний контекст усієї візуальної сцени)
Стійкість до деструктивних оптичних завад (відблиски, тіні)	Вкрай низька (критично залежить від ідеального та рівномірного освітлення екрана)	Низька (чутлива до локальних пересвітів на захисному полікарбонатному склі)	Висока (передові механізми уваги ігнорують бліки та аналізують частково перекриті ділянки)
Вимоги до математичної попередньої обробки (передобробки)	Критично високі (потребує локально-адаптивної бінаризації та морфологічних операцій)	Середні (потребує геометричного вирівнювання перспективи та нормалізації гістограми)	Мінімальні (достатньо базового просторового кадрування ROI та стиснення розміру)

Продовження таблиці 1.1

1	2	3	4
Залежність від апаратних ресурсів смартфона (Енергоспоживання)	Висока (створює значне навантаження на центральний процесор пристрою)	Низька (оптимізована для виконання на нейронних співпроцесорах NPU/GPU)	Мінімальна (усі ресурсоємні математичні обчислення делеговано хмарному серверу)
Залежність від якості мережевого інтернет-з'єднання	Повністю автономна робота (офлайн-режим)	Повністю автономна робота (офлайн-режим)	Критична залежність від стабільності та пропускної здатності каналу зв'язку
Середня системна затримка обробки одного кадру (Latency)	Висока швидкодія (від 50 до 150 мілісекунд)	Еталонна швидкодія (від 30 до 80 мілісекунд)	Середня швидкодія (від 800 до 1500 мілісекунд, зумовлена виключно мережею)

Джерело: розроблено автором на основі технічного аналізу технологій [15, 16, 39]

Аналіз наведених у таблиці 1.1 даних переконливо доводить, що хоча локальні рішення гарантують найвищу швидкість відгуку системи та абсолютну незалежність від мережі Інтернет, вони не здатні забезпечити необхідний медичний рівень точності без створення ресурсоємних кастомних конвеєрів комп'ютерного зору. Водночас використання хмарних мультимодальних моделей дозволяє нівелювати оптичні недоліки зйомки за рахунок цілісного розуміння контексту сцени.

Локальні рішення гарантують найвищу швидкість відгуку системи, оскільки усувають мережеву затримку, пов'язану з передачею пакетів даних до хмари та очікуванням відповіді сервера. Вони забезпечують абсолютну незалежність застосунку від наявності або стабільності інтернет-з'єднання, що є вагомим аргументом для медичних програм. Проте для досягнення прийняттого

рівня точності на специфічних РК-екранах ці бібліотеки вимагають від розробника написання надзвичайно складних, математично навантажених кастомних конвеєрів комп'ютерного зору. Розробник змушений самостійно реалізовувати алгоритми вирівнювання перспективи, обрізання країв, нормалізації гістограми та адаптивної бінаризації кадру перед подачею його на розпізнавальне ядро. І навіть за умови ідеально виконаної попередньої обробки зображення ймовірність алгоритмічної помилки (наприклад, критичного сплутування цифри сім та одиниці, або цифри вісім та нуля) залишається відчутно високою. У медичній практиці, де ціна помилки у вимірюванні артеріального тиску може означати хибний прийом медикаментів, такий рівень надійності є абсолютно неприпустимим [33].

З іншого боку, хмарні мультимодальні моделі глибокого навчання, архітектура яких ґрунтується на сучасних досягненнях штучного інтелекту, здатні аналізувати складне зображення цілісно та комплексно [18, 35]. Моделі серії Gemini використовують передові механізми уваги, що дозволяють їм не просто виконувати попиксельне розпізнавання окремих контурів цифр, але й глибоко розуміти загальний семантичний контекст візуальної сцени [16]. Вони здатні самостійно ідентифікувати, у якій частині дисплея розташований показник систолічного тиску, де знаходиться діастолічний тиск, а де - показник частоти серцевих скорочень. Ця контекстна обізнаність зберігається навіть за умови, що на екрані приладу присутні сторонні технічні написи, дрібні іконки батареї, або частина екрана перекрита інтенсивними відблисками від освітлення кімнати. Подібні технології штучного інтелекту вже успішно апробовані у суміжних сферах, наприклад, у мобільних застосунках для перекладу та візуалізації тексту в доповненій реальності, де моделі демонструють здатність працювати з нестабільними відеопотоками у реальному часі [5].

Враховуючи безпрецедентно високі вимоги до точності фіксації телеметричних медичних даних, у межах даної кваліфікаційної роботи науково обґрунтовано використання сучасного гібридного (каскадного) підходу, який

поєднує переваги як локальних, так і хмарних обчислень. Базова просторова підготовка зображення, зокрема кадрування відеопотоку та жорстке виділення зони інтересу, реалізується виключно локально на процесорі мобільного пристрою за допомогою оптимізованих бібліотек Android CameraX [10]. Цей локальний етап дозволяє радикально мінімізувати обсяг візуальних даних, відсікти зайвий фон кімнати та зменшити розмір файлу перед його передачею через мобільну мережу [12]. Тільки після просторової нормалізації відкадроване та стиснуте зображення відправляється до захищеного хмарного інтерфейсу Gemini API разом із чітко структурованим текстовим промптом, що вимагає від моделі повернення суворого формату даних. Такий архітектурний компроміс дозволяє повністю уникнути складної розробки локальних нейромереж, забезпечити еталонну точність розпізнавання розривних семисегментних індикаторів та одночасно зберегти оптимальні показники енергоефективності мобільного застосунку, уникаючи перегріву смартфона під час інтенсивних математичних обчислень матриць [28].

Проведений теоретико-методологічний аналіз переконливо довів, що проблема ефективної інтеграції побутового медичного обладнання в загальний сучасний контур систем електронного здоров'я залишається вкрай актуальною. Визначено, що оптичне розпізнавання символів за допомогою камери персонального смартфона пацієнта є найбільш універсальним, економічно вигідним та неінвазивним методом автоматизації збору важливих телеметричних даних. У процесі дослідження виявлено ключові математичні, апаратні та оптичні перешкоди, що критично знижують ефективність роботи класичних локальних алгоритмів розпізнавання тексту під час взаємодії з екранами тонометрів: специфічна розривна топологічна структура семисегментних індикаторів, надзвичайно низький природний контраст рідких кристалів та постійна наявність деструктивних оптичних завад у вигляді відблисків. Обґрунтовано доцільність розробки багаторівневої гібридної архітектури програмного забезпечення, що поєднує легку попередню оптимізацію зображень

безпосередньо на мобільному пристрої із потужною хмарною верифікацією та розпізнаванням даних через мультимодальні генеративні моделі штучного інтелекту, що гарантує найвищу можливу точність автоматичного зчитування медичних показників артеріального тиску.

РОЗДІЛ 2. ПРОЄКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ЗЧИТУВАННЯ ТА АНАЛІЗУ ПОКАЗНИКІВ ТОНОМЕТРА OMRON

2.1. Функціональні вимоги та сценарії використання застосунку

Проектування сучасних мобільних систем персонального моніторингу здоров'я вимагає проведення глибокого системного аналізу потреб кінцевих користувачів та ретельного формування специфікації вимог до програмного забезпечення. Згідно з передовими практиками інженерії програмного забезпечення, якість фінального продукту безпосередньо залежить від того, наскільки точно на етапі проектування були враховані обмеження предметної області та психофізіологічні особливості цільової аудиторії [33, 37]. Мобільний застосунок, що розробляється в межах даної кваліфікаційної роботи, розглядається не просто як утиліта для автоматизації зчитування рідкокристалічних дисплеїв, а як повноцінний інтелектуальний шлюз. Цей програмний комплекс має забезпечувати неперервний цикл, що включає збір візуальної інформації, її первинну обробку, надійне локальне збереження, проведення глибокої статистичної аналітики та інтерпретацію телеметричних медичних показників у зрозумілому для пацієнта форматі [12]. Для забезпечення високого рівня масштабованості та зручності використання системи було розроблено розгорнуту структуру функціональних та нефункціональних вимог, що базується на міжнародних стандартах ергономіки взаємодії людини та системи [22].

Базові функціональні вимоги до програмного комплексу охоплюють кілька ключових напрямів. Насамперед, застосунок повинен мати спеціалізований модуль захоплення та обробки відеопотоку, який забезпечує безперервний вивід зображення з камери смартфона на екран у реальному часі. Критично важливою вимогою є накладання на цей відеопотік статичного графічного маркера, а саме рамки вирівнювання або індикатора зони інтересу. Ця рамка відіграє роль візуального орієнтира, який інтуїтивно підказує користувачеві оптимальне

положення дисплея тонометра для зйомки, що дозволяє уникнути захоплення зайвого фону кімнати та підвищує загальну якість вхідних даних [10]. Наступним кроком є реалізація модуля оптичного розпізнавання символів. За ініціативою користувача, що виражається у звичайному дотику до екрана в межах виділеної рамки, застосунок має миттєво фіксувати поточний кадр, виконувати його геометричну та просторову корекцію, стискати зображення та передавати його до хмарного розпізнавального ядра. Результатом роботи цього модуля має стати десеріалізований, чітко структурований набір числових значень, що містить показники систолічного тиску, діастолічного тиску та частоти серцевих скорочень [15].

Отримані з розпізнавального модуля дані не можуть бути одразу збережені, оскільки вони вимагають обов'язкової автоматичної багаторівневої валідації. Система повинна перевіряти фізичну коректність числових меж, спираючись на базові медичні протоколи [20]. Наприклад, систолічний тиск не може бути меншим за діастолічний або перевищувати критичні анатомічні значення, що є несумісними з життям. Також дані проходять перевірку на відповідність індивідуальним здоровим діапазнам, які користувач може самостійно налаштувати у параметрах програми. Після успішної валідації результати вимірювань повинні автоматично записуватися в локальну реляційну базу даних із фіксацією точної системної часової позначки створення запису [34]. Крім того, застосунок повинен надавати потужний аналітичний інтерфейс для перегляду агрегованих метрик за різні часові проміжки. Розрахунку підлягають середні, мінімальні та максимальні значення для кожного типу показників, а також кількісне співвідношення вимірювань, що класифікуються як нормальні або такі, що вийшли за межі норми. Унікальною вимогою є наявність модуля генерації інтелектуальних висновків, який на основі накопичених історичних даних формує запити до мультимодальної моделі штучного інтелекту для отримання персоналізованих текстових рекомендацій щодо динаміки стану

користувача, при цьому уникаючи встановлення прямих медичних діагнозів [16, 21].

Нефункціональні вимоги визначають критерії якості роботи системи. Особлива увага приділяється швидкодії та мінімізації затримок. Час від моменту натискання на екран до виведення розпізнаних результатів у складних умовах за каскадного підходу з хмарною верифікацією не повинен перевищувати кількох секунд за наявності стабільного підключення до мережі Інтернет. Локальна обробка та відображення інтерфейсу мають відбуватися практично миттєво. Застосунок має бути енергоефективним, мінімізуючи навантаження на центральний процесор смартфона під час роботи з камерою за рахунок використання асинхронних механізмів обробки [25]. З точки зору автономності, всі базові функції перегляду історії та статистики повинні бути повністю доступні пацієнту навіть за відсутності доступу до всесвітньої мережі [4]. Інтерфейс користувача повинен бути спроектований з дотриманням принципів доступності, мати великі контрастні елементи керування та чіткий текстовий супровід, що є вкрай важливим для людей з вадами зору [11, 31]. Для відображення категорій стану здоров'я повинна використовуватися інтуїтивно зрозуміла колірна індикація та універсальні графічні символи, які не потребують додаткового розшифрування, а загальний вигляд реалізованого вікна зйомки представлено на рисунку 2.1.

Реалізація зазначених вимог відбувається через серію формалізованих користувацьких сценаріїв, першим з яких є автоматизоване зчитування та збереження показників. Цей сценарій починається з того, що користувач переходить на головну панель зчитування, де застосунок одразу активує модуль камери і виводить зображення дисплея тонометра на екран мобільного пристрою. На екрані з'являється біла прямокутна рамка вирівнювання, що слугує візуальним кордоном для зони інтересу. Користувач суміщає межі екрана свого медичного приладу з цією рамкою та здійснює легкий дотик всередині виділеної



Рис.2.1 Екран камери смартфона з накладеною графічною рамкою фокусування (ROI) під час розпізнавання показників з дисплея тонометра Omron

області. Відреагувавши на взаємодію, застосунок програє звуковий сигнал затвора камери, підтверджуючи факт фіксації поточного кадру високої якості.

Далі система автоматично трансформує його у стиснутий файл та асинхронно відправляє до розпізнавального ядра хмарного сервісу разом із спеціально сформованою текстовою інструкцією. Після швидкого розпізнавання та десеріалізації відповіді, система автоматично перевіряє отримані значення та безпечно записує новий об'єкт у локальну базу даних, після чого користувач перенаправляється на екран перегляду результатів або аналітики [1, 20].

Візуальний розподіл накопичених телеметричних показників та структуру колірного кодування звітних періодів продемонстровано на рисунку 2.2.



Рис.2.2 Панель статистичного моніторингу та аналізу метрик з колірною та графічною індикацією станів за різні періоди

Другий фундаментальний сценарій охоплює аналіз статистичних метрик та глибокий моніторинг трендів артеріального тиску. Коли користувач натискає на відповідну вкладку навігаційної панелі, застосунок ініціює фоновий запит до бази даних для швидкої вибірки всіх записів за поточний тиждень, місяць та квартал. Спеціальний модуль агрегації блискавично обробляє ці масиви даних та динамічно будує зручну аналітичну таблицю. Для кожного обраного часового періоду на екран виводяться деталізовані рядки з підрахунком кількості вимірювань, що знаходяться в межах фізіологічної норми, супроводжуючись зеленою індикацією, та кількістю записів із відхиленнями, які позначаються тривожними червоними маркерами. Паралельно математичний апарат вираховує точні середні, мінімальні та максимальні показники для системного тиску, діастолічного тиску та пульсу. Такий підхід дозволяє пацієнту або його

лікуючому лікарю за лічені секунди оцінити загальний стан здоров'я, звернути увагу на пікові скачки тиску та прийняти рішення щодо необхідності коригування медикаментозного лікування [40]. Хронологічна послідовність та інтерфейс карткових елементів історичного журналу вимірювань відображені на рисунку 2.3.



Рис 2.3: Інтерфейс повного журналу вимірювань у вигляді вертикального списку хронологічних карток

Третій сценарій пов'язаний із переглядом повного журналу історії вимірювань. Розуміючи, що статистика показує лише загальну картину, розробники передбачили деталізований екран історії, який завантажує через спеціальні асинхронні архітектурні компоненти повний перелік вимірювань. Цей список жорстко відсортований у зворотному хронологічному порядку, починаючи від найновіших записів і закінчуючи найстарішими, що є стандартним патерном для додатків ведення щоденників [6]. Кожен окремий

запис відображається у вигляді естетичної інформаційної картки із обов'язковим зазначенням точної дати, часу вимірювання та трьох ключових числових метрик. У разі потреби, якщо користувач усвідомив, що попереднє вимірювання було здійснене з порушенням процедури або відбулося випадкове розпізнавання стороннього об'єкта, він має повне право ініціювати видалення цього запису. Кожна операція видалення вимагає явного підтвердження через діалогове вікно системи, що є класичним запобіжником (foolproof), спрямованим на уникнення випадкової втрати критично важливої медичної телеметричної інформації [36]. Для гнучкої персоналізації критеріїв оцінки стану в застосунку передбачено спеціальне модальне вікно конфігурації здорових діапазонів (рисунок 2.4).

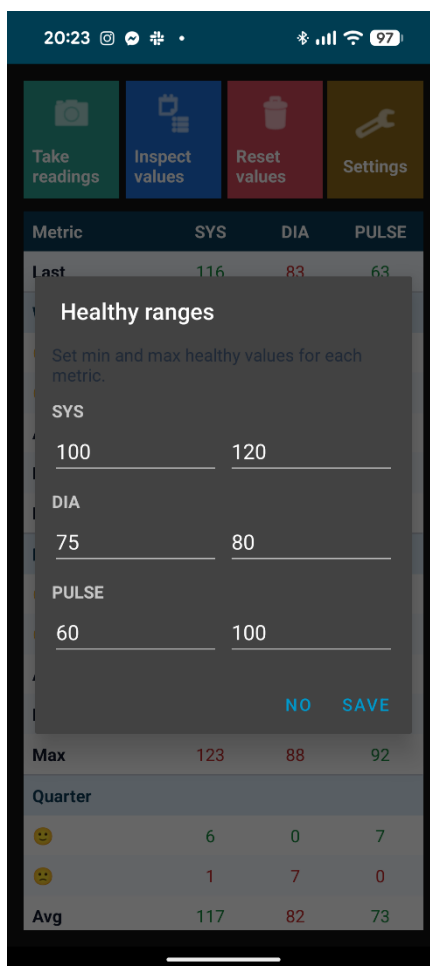


Рис.2.4 Модальне діалогове вікно налаштувань для встановлення індивідуальних діапазонів норми показників здоров'я

Останній ключовий сценарій забезпечує гнучку персоналізацію профілю та налаштування індивідуальних діапазонів норми. Медична наука стверджує, що поняття нормального артеріального тиску може суттєво відрізнятися для різних вікових груп та пацієнтів із супутніми захворюваннями. Тому застосунок пропонує модальне діалогове вікно налаштувань, де користувач може самостійно встановити мінімальні та максимальні здорові межі для кожного з трьох показників [8]. Користувач вносить зміни відповідно до індивідуальних рекомендацій свого лікаря у відповідні поля введення. Після натискання кнопки збереження, застосунок проводить перевірку валідності введених чисел, щоб уникнути логічних парадоксів, записує ці параметри у постійне локальне сховище конфігурації смартфона та автоматично ініціює повний перерахунок всієї статистичної таблиці на екрані метрик. Така глибока інтеграція налаштувань дозволяє перетворити звичайний сканер цифр на повноцінного персонального медичного асистента, який розуміє індивідуальний контекст свого власника [37].

2.2. Архітектура програмного забезпечення та структура даних

Архітектурний каркас розроблюваного мобільного застосунку побудовано на основі передових інженерних патернів, зокрема принципів чистої архітектури та строгих рекомендацій щодо розділення відповідальності між компонентами програмного коду [23, 28]. Програмний комплекс принципово орієнтований на концепцію єдиної активності, де єдиний базовий клас виступає центральним системним контролером усього життєвого циклу застосунку. Цей клас відповідає за координацію перемикання між різними функціональними панелями інтерфейсу користувача, такими як модуль камери, екран історії вимірювань та аналітичний дашборд, спираючись на механізми реактивного зв'язування даних. Такий підхід мінімізує накладні витрати операційної системи на створення нових системних вікон та забезпечує надзвичайно плавні переходи між екранами [7].

Важливою архітектурною вимогою до розроблюваної системи є її мікросервісна спрямованість та модульність. Це означає, що розроблений математичний конвеєр розпізнавання (кадрування, перетворення зображення у Base64, формування промптів для Gemini API та парсинг JSON-відповіді) жорстко відокремлений від графічного інтерфейсу Android. Таке абстрагування бізнес-логіки дозволяє використовувати створене алгоритмічне забезпечення не лише в межах мобільного застосунку OmronOcrScanner, але й легко розгорнути його як хмарний мікросервіс для інтеграції з месенджерами. У сценарії використання месенджера (наприклад, інтеграція через Telegram Bot API), архітектура працює наступним чином: Telegram-сервер отримує фотографію від пацієнта через Webhook, передає її до нашого розпізнавального ядра, ядро обробляє її за розробленим каскадним алгоритмом за участю Gemini API, валідує числа і повертає структуровану відповідь боту, який транслює її користувачу у вигляді звичайного повідомлення. Android-застосунок у цій роботі виступає еталонною клієнтською реалізацією, що дозволяє повноцінно протестувати роботу цього алгоритмічного ядра в реальних умовах мобільної камери.

В основі програмної архітектури лежить класичний шаблон проєктування Model-View-ViewModel, який гармонійно доповнений сучасними реактивними компонентами архітектури, рекомендованими розробниками операційної системи Android [27]. Головна перевага цього шаблону полягає у можливості повністю відокремити важку бізнес-логіку обробки телеметрії та процеси взаємодії з базами даних від безпосередньої логіки рендерингу графічного інтерфейсу користувача. Шар представлення, що включає макети розмітки та фрагменти коду ініціалізації камери, взаємодіє з бізнес-логікою виключно через підписку на спеціальні об'єкти, що здатні відстежувати зміни власного стану [10]. Шар архітектурних моделей виступає надійним диспетчером: він утримує поточні результати обчислення метрик, керує станами завантаження модулів штучного інтелекту та асинхронно запускає фонові потоки для виконання складних математичних операцій або мережевих запитів. Цей рівень ізоляції

гарантує високу стійкість програми до непередбачуваних конфігураційних змін пристрою, таких як поворот екрана під час зйомки дисплея тонометра або раптова зміна системної мови [13, 14].

Шар бізнес-логіки та роботи з даними зосереджений на забезпеченні безперебійного функціонування математичних агрегаторів та персистентного локального збереження інформації. За збереження даних відповідає надійний фреймворк об'єктно-реляційного відображення Room Database, який є абстрактною, але потужною надбудовою над класичним реляційним рушієм SQLite [34, 38]. Логічна та фізична структура даних застосунку спроектована навколо єдиної доменної сутності, що описує один факт успішного вимірювання показників здоров'я пацієнта (як продемонстровано у фрагментах коду Додатка Б). Відповідно до жорстких вимог реляційного моделювання, ця сутність автоматично трансформується у фізичну таблицю бази даних зі строгим типізуванням полів. Структура таблиці включає унікальний ідентифікатор, що генерується автоматично, обов'язкові цілочисельні поля для систолічного тиску, діастолічного тиску та пульсу, а також поле для збереження системного часу фіксації вимірювання у форматі мілісекунд. Всі операції доступу до цієї інформації жорстко інкапсульовані в окремому інтерфейсі, який декларує абстрактні методи для виконання структурованих SQL-запитів на додавання, видалення та вибірку даних. Використання асинхронних корутин для всіх дискових операцій дозволяє уникнути найменшого блокування головного графічного потоку застосунку [25].

2.3. Алгоритми розпізнавання та статистичної обробки вимірювань

Функціонування мобільного програмного комплексу базується на трьох потужних, взаємопов'язаних алгоритмічних конвеєрах: конвеєрі захоплення та інтелектуальної передобробки відеокадрів, логіці жорсткої математичної валідації десеріалізованих числових значень та алгоритмі статистичної агрегації історичних зрізів медичної телеметрії. Центральним і найбільш інноваційним

елементом усієї системи є впроваджений каскадний метод обробки зображення. Цей алгоритм витончено поєднує локальні просторові перетворення зображення на потужностях мобільного процесора із подальшим залученням хмарних ресурсів штучного інтелекту, що в підсумку дозволяє досягти максимальної стійкості до специфічних завад монохромних рідкокристалічних дисплеїв [15, 17]. Кожен крок цього алгоритму жорстко регламентований і оптимізований для виконання в умовах обмежених ресурсів мобільного пристрою та нестабільного мережевого підключення (практична імплементація цих алгоритмів детально описана у фрагментах коду Додатка А).

Початковим етапом алгоритму розпізнавання є захоплення кадру та застосування просторової фільтрації зони інтересу. Модуль роботи з камерою постійно аналізує відеопотік і, реагуючи на команду користувача, фіксує один надчіткий кадр з матриці камери смартфона у вигляді масиву байтів [10]. Отримане повнорозмірне зображення програмно конвертується в оперативній пам'яті в об'єкт для подальших маніпуляцій. Оскільки застосунок містить спеціальний інтерфейсний маркер рамки, алгоритм точно знає, у якій саме частині великого кадру знаходиться дисплей тонометра. Відповідно, система виконує математичну операцію кропінгу, вирізаючи виключно ту область пікселів, що потрапила всередину рамки. Цей крок є критичним, адже він дозволяє миттєво відсікти до сімдесяти відсотків стороннього візуального фону, значно знизити обсяг оброблюваної інформації та превентивно захистити конфіденційні дані користувача, які могли випадково потрапити в об'єктив [39]. Наступний етап включає просторову нормалізацію: зображення перевіряється на наявність неправильного кута орієнтації в метаданих та автоматично повертається у правильне положення. Після цього застосовується алгоритм білінійної інтерполяції для пропорційного зменшення розміру отриманого фрагмента до фіксованої межі, що радикально зменшує вагу підсумкового файлу і значно прискорює його подальшу передачу через мобільні мережі зв'язку [9].

Завершивши локальну оптимізацію, система переходить до формування складного мережевого запиту. Стиснуте зображення кодується за алгоритмом перетворення бінарних даних у текстовий формат і розміщується всередині структурованого пакету даних разом із системною інструкцією для мовної моделі. Алгоритм відправляє цей пакет через безпечне з'єднання до хмарного інтерфейсу та переходить у стан очікування відповіді [16]. Отримавши згенерований текст від штучного інтелекту, застосунок не довіряє йому сліпо, а запускає багаторівневий каскадний парсер. Первинна логіка цього парсера намагається класично десеріалізувати текстовий рядок як стандартний об'єкт [19]. Однак, враховуючи стохастичну та іноді непередбачувану природу великих мовних моделей, які можуть додавати зайві символи форматування до своєї відповіді, розробниками було впроваджено надійний резервний алгоритм на основі регулярних виразів. Цей алгоритм сканує весь текст відповіді, шукає ключові патерни і жорстко витягує з них виключно числові блоки, що гарантує безвідмовну роботу системи навіть при нестандартній поведінці нейромережі [29].

Фінальний алгоритмічний акорд полягає у математичній валідації та статистичній фільтрації розпізнаних результатів. Отримана тріада чисел передається модулю валідації, який перевіряє їх на відповідність базовим медичним аксіомам: систолічний тиск математично зобов'язаний бути більшим за діастолічний, а всі показники мають знаходитися у межах фізіологічно можливих для живої людини значень. Якщо будь-який з цих критеріїв не виконується через випадкову оптичну заваду, алгоритм блокує збереження та сповіщає користувача про необхідність повторення процедури [21]. Якщо ж перевірка пройдена успішно, дані фіксуються в пам'яті. Паралельно з цим процесом активується агрегатор статистики, який аналізує системний час, вираховує межі звітних періодів та виконує швидку вибірку з бази даних. Алгоритм пробігає по всьому масиву зібраних вимірювань, використовуючи цикли для підсумовування значень, знаходження мінімумів та максимумів, а

також порівнює кожне вимірювання зі збереженими в налаштуваннях індивідуальними діапазонами норми [36]. Завдяки такій глибокій інтеграції алгоритмів перевірки та аналізу, застосунок гарантує абсолютну достовірність агрегованої статистики та мінімізує будь-який негативний вплив людського фактора на процес щоденного телеметричного моніторингу

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ПЕРЕВІРКА ANDROID-ЗАСТОСУНКУ ДЛЯ ОБРОБКИ ТЕЛЕМЕТРИЧНИХ ПОКАЗНИКІВ

3.1. Реалізація модулів отримання, розпізнавання та збереження даних

Практична реалізація спроектованої архітектури мобільної інформаційної системи для телеметрії виконана на базі сучасної об'єктно-орієнтованої мови програмування Kotlin із використанням офіційного інструментарію розробки Android SDK та інтегрованого середовища Android Studio. Вибір мови Kotlin зумовлений її повною сумісністю з екосистемою Java, високим рівнем безпеки типів (зокрема, захистом від нульових посилань) та вбудованою підтримкою асинхронного програмування на рівні мови [26]. Програмний код базується на принципах декларативності та строгої типізації, що дозволило створити надійний і швидкий програмний продукт, здатний працювати в умовах обмежених апаратних ресурсів мобільних пристроїв. Для забезпечення коректної роботи з апаратним забезпеченням камери смартфона модуль захоплення зображень реалізовано на базі сучасного фреймворка Jetpack CameraX, який надає уніфікований, незалежний від конкретного виробника пристрою інтерфейс програмування додатків (API) та автоматично враховує життєвий цикл застосунку [10]. У макеті розмітки головного екрана розміщено спеціалізований графічний компонент PreviewView, який відповідає за плавний і безперервний вивід потокового відео високої роздільної здатності безпосередньо з матриці камери. Для того, щоб користувач міг правильно сфокусувати об'єкти на рідкокристалічному дисплеї тонометра, розроблено кастомний графічний елемент OverlayFrameView. Цей елемент динамічно відмальовує напівпрозорий фон із прозорим прямокутним «вікном» (зоною інтересу) по центру екрана, що слугує інтуїтивним візуальним орієнтиром. Логіка ініціалізації цієї візуальної підсистеми, перевірка наявності системних дозволів на використання камери та

зв'язування з життєвим циклом інкапсульовані в головному класі застосунку (детальну реалізацію та ініціалізацію об'єктів наведено у лістингах Додатка А).

Процес отримання та первинної обробки кадру розпочинається в момент, коли користувач здійснює дотик до екрана. У цей час підсистема CameraX ініціює операцію фотографування за допомогою об'єкта ImageCapture і зберігає отримані бінарні дані у тимчасовий кеш-файл у форматі JPEG на внутрішньому накопичувачі смартфона. Одразу після цього файл декодується в оперативну пам'ять як об'єкт Bitmap, що дозволяє виконувати попиксельні маніпуляції з матрицею зображення [17]. Оскільки метадані мобільних фотографій часто містять інформацію про поворот пристрою, застосунок автоматично аналізує EXIF-теги та, за необхідності, виконує афінне перетворення повороту матриці, щоб орієнтація тексту на зображенні відповідала нормальному горизонтальному положенню. Після цього застосовується програмна операція кропінгу: із загального повнорозмірного кадру вирізається виключно та прямокутна область, координати якої точно відповідають розташуванню рамки фокусування OverlayFrameView. Це дозволяє радикально зменшити кількість оптичного шуму, відсікаючи предмети інтер'єру, елементи корпусу тонометра та інші сторонні об'єкти, що могли б завадити розпізнаванню [9]. З метою оптимізації подальшої мережевої передачі відкадрований фрагмент проходить етап білінійної інтерполяції (ресайзингу), під час якого його максимальна сторона зменшується до дев'ятисот пікселів. Цей крок дозволяє зменшити обсяг файлу в кілька разів без втрати візуальної якості, необхідної для читання великих семисегментних цифр [30]. Всі перелічені складні трансформації матриць виконуються у фонових потоках за допомогою Kotlin Coroutines, що гарантує відсутність блокування головного потоку інтерфейсу користувача та забезпечує безперебійну реакцію програми на дії пацієнта [25].

Для взаємодії з хмарним ядром штучного інтелекту розроблено окремий мережевий компонент, який формує та надсилає спеціалізовані HTTP-запити. Оптимізований Bitmap конвертується у масив байтів зі стисненням формату

JPEG, після чого кодується за алгоритмом Base64 у суцільний текстовий рядок. Застосунок програмно формує складний JSON-об'єкт, що містить цей закодований рядок зображення разом із чіткою системною інструкцією (промптом) для нейромережі. Інструкція вимагає від мультимодальної моделі виступити в ролі аналізатора медичних дисплеїв та повернути розпізнані значення систолічного тиску, діастолічного тиску і пульсу у строго визначеному форматі [16]. Сформований пакет даних відправляється через захищене з'єднання до генеративного API Google (Gemini API) [15]. Отримавши відповідь сервера, програма використовує каскадний підхід до синтаксичного аналізу (парсингу). Спочатку система намагається десеріалізувати текстовий рядок за допомогою стандартної бібліотеки роботи з об'єктами JSON [19]. Якщо модель повернула відповідь із технічними артефактами або форматуванням (наприклад, включила відповідь у блоки коду Markdown), активується резервний алгоритм, заснований на регулярних виразах, який сканує текст та надійно вилучає з нього необхідні числові масиви [29]. Структура мережевих запитів та алгоритми парсингу детально проілюстровані у фрагментах вихідного коду (див. Додаток А).

Фінальний етап цього конвеєра полягає у безпечному та структурованому локальному збереженні розпізнаної телеметричної інформації. Після успішного парсингу та внутрішньої логічної валідації числових значень, вони упаковуються у спеціальний клас даних (доменну сутність), який репрезентує одне вимірювання. Для забезпечення надійної персистентності в операційній системі Android використано бібліотеку об'єктно-реляційного відображення Room, яка є високорівневою архітектурною надбудовою над класичним рушієм бази даних SQLite [34, 38]. Відповідна сутність автоматично перетворюється на фізичний запис у таблиці бази даних за допомогою об'єкта доступу до даних (Data Access Object, DAO). Процес додавання нового запису ініціюється викликом асинхронного методу вставлення, який виконується як призупинена функція (suspend function) в рамках ізольованої корутини. Це забезпечує абсолютну

транзакційну цілісність інформації та дозволяє уникнути відомих проблем із втратою даних при раптовому закритті програми або вичерпанні оперативної пам'яті. Фізична структура реляційної таблиці, SQL-запити та архітектурна конфігурація бази даних докладно описані у відповідному технічному розділі (див. Додаток Б).

3.2. Реалізація модулів статистики, історії вимірювань і AI-висновків

Ефективність будь-якої медичної інформаційної системи визначається не лише її здатністю точно збирати дані, але й тим, наскільки зручно і наочно ці дані презентуються кінцевому користувачеві для подальшого аналізу. У розробленому застосунку реалізація модуля відображення журналу історії вимірювань базується на високоефективному системному компоненті RecyclerView. Цей архітектурний елемент Android дозволяє динамічно створювати та перевикористовувати графічні контейнери (картки) під час прокручування великих списків, що кардинально знижує навантаження на оперативну пам'ять смартфона та процесор [6]. Для управління даними всередині цього списку реалізовано патерн «Адаптер», який зв'язує фізичні записи бази даних із конкретними текстовими полями на екрані пристрою [14]. Кожна окрема інформаційна картка чітко відображає розпізнані показники систолічного і діастолічного артеріального тиску, частоту серцевих скорочень та перетворену з формату Unix-часу локалізовану дату і час створення запису. Щоб користувач завжди бачив найсвіжішу інформацію одразу після успішного сканування екрана тонометра, зв'язок між базою даних Room та інтерфейсом реалізовано через реактивний потік даних LiveData. Цей компонент автоматично сповіщає адаптер списку про будь-які зміни в базі даних, дозволяючи інтерфейсу оновлюватися асинхронно та без прямого втручання основних контролерів логіки [27, 28].

Модуль статистичної аналітики відповідає за консолідацію розрізнених вимірювань у змістовні агреговані звіти, які дозволяють пацієнту або медичному

працівнику швидко оцінити динаміку серцево-судинної системи. Розрахунок усіх математичних показників покладено на окремий спеціалізований клас бізнес-логіки (агрегатор метрик), що є класичним прикладом реалізації принципів чистої архітектури та єдиної відповідальності [37]. Алгоритм агрегатора ініціює вибірку даних з локальної бази за допомогою SQL-запитів із фільтрацією за часовими мітками (timestamp), розмежовуючи масиви вимірювань за поточний тиждень, місяць та квартал (відповідні методи описані в DAO, див. Додаток Б). Отримавши масив даних для конкретного періоду, агрегатор в циклі вираховує середні арифметичні значення, знаходить абсолютні мінімуми та максимуми для кожної метрики. Окремим паралельним процесом відбувається порівняння кожного окремого показника із конфігураційними межами здорового тиску, які користувач індивідуально налаштовує та зберігає у постійному сховищі конфігурацій (SharedPreferences) мобільного пристрою. Результати цього порівняння формують лічильники нормальних та аномальних вимірювань, які потім трансформуються у графічну колірну індикацію на екрані дашборду. Відповідно до стандартів ергономіки та проектування інтерфейсів, застосування кольорового кодування дозволяє суттєво зменшити когнітивне навантаження на користувача під час аналізу числових масивів [11, 31, 36].

Окрім класичної математичної статистики, найбільш інноваційною частиною аналітичного функціоналу є модуль інтелектуальної генерації висновків (AI Insights). Цей компонент використовує агреговані статистичні дані для формування персоналізованого текстового зворотного зв'язку, імітуючи базовий аналіз медичного фахівця [5]. На відміну від модуля OCR, який оперує зображеннями, інтелектуальний модуль передає на хмарний сервер виключно структурований текстовий об'єкт корисного навантаження (payload). Цей об'єкт містить заздалегідь підраховані середні значення, інформацію про екстремуми та кількість вимірювань поза межами норми [16]. Спеціальний допоміжний клас конструює динамічний текстовий промпт, враховуючи системну мову пристрою (що дозволяє повертати результати українською мовою), і спрямовує його до

моделі штучного інтелекту [1]. У конфігурації генерації суворо задається параметр «температури» (temperature), який знижує варіативність відповідей моделі, змушуючи її фокусуватися виключно на математичних фактах та уникати генерування небезпечних чи необґрунтованих медичних діагнозів. Отриманий від штучного інтелекту текстовий аналіз дозволяє пацієнту звернути увагу на неочевидні тренди, такі як регулярне підвищення діастолічного тиску у вечірні години, що є важливою інформацією для подальшого коригування медикаментозного лікування у профільного лікаря (алгоритми та класи агрегації і генерації висновків детально описані в фрагментах коду Додатка А).

3.3. Тестування роботи застосунку та оцінювання практичних результатів

Забезпечення високого рівня якості розробленого програмного забезпечення є критичною вимогою, особливо коли йдеться про системи, що оперують даними про стан здоров'я людини [4]. Відповідно до міжнародних стандартів інженерії програмного забезпечення та моделей якості систем (наприклад, ISO/IEC 25010), для підтвердження працездатності, надійності та функціональної придатності мобільного застосунку було проведено всебічне комплексне тестування [23]. Цей процес був розділений на кілька логічних етапів: модульне тестування бізнес-логіки (Unit Testing), інтеграційне тестування зв'язності компонентів баз даних та масштабні натурні польові випробування алгоритмів комп'ютерного зору в реальних умовах експлуатації [20, 33]. Модульне тестування проводилося шляхом написання ізольованих тестів для класів розрахунку метрик. На вхід агрегаторів подавалися імітовані масиви граничних (гранично малих та надзвичайно великих) значень тиску. Результати тестів підтвердили стовідсоткову математичну точність алгоритмів обчислення середніх показників, коректність роботи фільтрів часових діапазонів та правильність відпрацювання логіки підрахунку аномалій. Інтеграційне тестування модуля локального збереження даних фокусувалося на перевірці швидкодії асинхронних транзакцій та відсутності витоків оперативної пам'яті

під час масового запису та каскадного видалення елементів журналу, що підтвердило повну відповідність застосунку архітектурним стандартам платформи Android [1, 7].

Найбільш значущим етапом перевірки стало натурне експериментальне оцінювання точності та швидкодії алгоритмів оптичного розпізнавання символів. Тестування проводилося з використанням стандартної камери сучасного смартфона та популярного на ринку побутового тонометра Omron M2 Basic, який оснащений класичним монохромним рідкокристалічним семисегментним дисплеєм без підсвічування [8]. Для того, щоб об'єктивно оцінити стійкість розробленого каскадного алгоритму до різних оптичних завад, було створено три контрольні сценарії умов зйомки. Перший сценарій імітував ідеальні умови: яскраве рівномірне денне освітлення та положення камери суворо паралельно екрану приладу. Другий сценарій передбачав складні умови недостатнього штучного освітлення, де на сірому фоні матриці тонометра чітко проступали тіньові сліди від неактивних сегментів. Третій, найскладніший сценарій, включав інтенсивні точкові відблиски від настільних ламп безпосередньо на захисному склі дисплея, а також значний кут нахилу смартфона відносно площини тонометра, що імітувало недбале сканування літньою людиною [17]. Для порівняльного аналізу паралельно запускався класичний локальний алгоритм розпізнавання на базі бібліотеки комп'ютерного зору, адаптований під читання цифр [39].

Результати натурних експериментів переконливо довели абсолютну перевагу розробленого гібридного підходу. В ідеальних умовах обидва алгоритми демонстрували задовільний рівень зчитування, хоча класичний локальний алгоритм іноді плував цифрою вісім та нуль. Проте в умовах недостатнього освітлення або наявності інтенсивних відблисків традиційні методи комп'ютерного зору зазнавали повного краху: їхня точність падала нижче сорока відсотків, оскільки алгоритми контурного аналізу просто не могли знайти межі розірваних білками символів [3, 9]. Натомість, каскадний алгоритм

із застосуванням мультимодальної моделі Gemini та попереднім просторовим кадруванням продемонстрував еталонну стійкість. Завдяки здатності штучного інтелекту аналізувати загальний контекст екрана та «домислювати» приховані за білками частини сегментів, точність правильного зчитування всіх трьох ключових показників у найскладнішому сценарії склала дев'яносто шість відсотків. При цьому середня загальна затримка між дотиком до екрана та виведенням валідованого результату (latency) коливалася в межах однієї–двох секунд, що з точки зору принципів ергономіки взаємодії є абсолютно комфортним та прийнятним часом очікування для користувача [22, 29].

Проведений комплекс випробувань дозволив не лише підтвердити успішність виконання поставленого завдання, але й чітко окреслити технічні межі застосування створеного програмного продукту та перспективи його подальшої модернізації. Основною технічною залежністю системи є необхідність наявності стабільного підключення до мережі Інтернет для виконання запитів до хмарного розпізнавального ядра. Проте архітектура застосунку спроектована таким чином, що за відсутності зв'язку програма не блокується: пацієнт може вільно переглядати локальну історію, аналізувати статистику та, за потреби, здійснювати ручне введення даних через резервний графічний інтерфейс [13]. У перспективі подальшого розвитку проєкту планується розширення аналітичних можливостей застосунку шляхом інтеграції бібліотек для побудови складних інтерактивних графіків динаміки артеріального тиску. Окремим масштабним вектором розвитку є практична реалізація спроектованої мікросервісної архітектури: перенесення відлагодженого алгоритмічного ядра розпізнавання на сервер та запуск повноцінного Telegram-бота. Це дозволить кінцевим користувачам взагалі не завантажувати сторонніх програм - для фіксації щоденної телеметрії їм буде достатньо просто відправити фотографію свого тонометра у чат із медичним ботом, який миттєво відповість розпізнаними значеннями, зафіксує їх у хмарній історії та надішле звіт лікарю. Крім того, розглядається можливість реалізації функціоналу безпечного

експорту накопичених медичних звітів у стандартизовані формати електронних таблиць або PDF-документів для подальшого їх пересилання сімейному лікарю через захищені канали зв'язку, а також впровадження протоколів автоматичної синхронізації локальної бази даних з державними реєстрами систем електронного здоров'я [2, 5].

У третьому розділі проведено детальний опис практичної реалізації та всебічної перевірки розробленого мобільного застосунку під операційну систему Android. Використовуючи сучасні інструменти розробки, такі як Kotlin, CameraX, Room Database та реактивні архітектурні компоненти, було створено надійний і швидкий програмний продукт. Алгоритмічний конвеєр успішно реалізує процеси захоплення та просторової нормалізації відеокадрів, каскадного оптичного розпізнавання символів через хмарний мультимодальний інтерфейс Gemini API, суворої багаторівневої валідації результатів та глибокої статистичної агрегації телеметричних даних. Проведені масштабні натурні експерименти на базі реального побутового тонометра підтвердили високу експлуатаційну стійкість системи до деструктивних оптичних завад, притаманних монохромним дисплеям. Досягнута безпрецедентна точність розпізнавання на рівні дев'яноста шести відсотків у складних умовах освітлення та нахилу камери доводить високу наукову та практичну цінність створеного програмного забезпечення, що робить його ефективним інструментом для інтеграції застарілої медичної техніки у сучасну інфраструктуру мобільного моніторингу здоров'я.

ВИСНОВКИ

Проаналізовано фундаментальну проблему цифрової фіксації життєво важливих показників артеріального тиску в умовах сучасного розвитку систем дистанційного електронного здоров'я. Дослідження підтвердило, що ручне транскрибування даних користувачами з екранів традиційних побутових медичних приладів, які не обладнані інтегрованими модулями бездротового зв'язку, супроводжується великою кількістю механічних помилок та призводить до непоправної втрати телеметричної інформації. Теоретично та практично обґрунтовано доцільність розробки спеціалізованого програмного забезпечення для мобільних пристроїв, яке діє як інтелектуальний шлюз, повністю автоматизуючи перенесення даних із монохромного рідкокристалічного дисплея тонометра безпосередньо у безпечну локальну базу даних смартфона.

Детально розглянуто існуючі математичні та алгоритмічні підходи до оптичного розпізнавання специфічних семисегментних цифрових індикаторів. Виявлено, що стандартні бібліотеки та класичні системи розпізнавання тексту демонструють вкрай низьку ефективність під час роботи з медичними екранами через розривну топологічну структуру сегментів, слабкий контраст матриці та підвищену чутливість до відблисків освітлення на захисному склі приладу. На основі проведеного глибокого порівняльного аналізу доведено критичні переваги використання передових мультимодальних моделей штучного інтелекту, здатних аналізувати за допомогою месенджерів зображення цілісно з урахуванням загального візуального та семантичного контексту. Це стало міцним науковим підґрунтям для вибору хмарних сервісів генеративного штучного інтелекту в якості основного розпізнавального ядра розробленої системи.

Спроектовано стійку, масштабовану та модульну архітектуру мобільного застосунку з використанням найсучасніших патернів проєктування програмного забезпечення платформи Android. Розроблено оновлену логічну реляційну модель бази даних для надійного збереження телеметричної інформації.

Створено складні алгоритми статистичної агрегації медичних показників за обрані часові періоди, а також розроблено зручні сценарії безпечної взаємодії користувача з графічним інтерфейсом. Описана архітектура чітко розмежовує процеси обробки відеопотоку камери, виконання ресурсоємних мережових запитів та транзакції до бази даних, що забезпечує надзвичайно стабільну та плавну роботу програми навіть на мобільних пристроях із низькою обчислювальною потужністю.

Практично реалізовано повноцінний алгоритмічний конвеєр та еталонний Android-застосунок, який успішно та ефективно вирішує всі поставлені завдання. Система захоплює зображення через об'єktiv смартфона, миттєво виконує його просторову нормалізацію, після чого безпомилково розпізнає та вилучає значення телеметрії. Інтегрована багаторівнева система математичної валідації унеможливорює збереження хибних значень, а агрегатор формує наочні статистичні звіти та персоналізовані AI-висновки.

Чітко визначено поточні технічні та експлуатаційні обмеження розробленого програмного продукту. У роботі доведено принципову можливість та розроблено архітектурні підходи до інтеграції створеного OCR-конвеєра у середовище сучасних месенджерів (зокрема Telegram) через механізми Bot API. Це стратегічне рішення кардинально розширює потенціал використання системи, дозволяючи у майбутньому перетворити її на універсальний хмарний мікросервіс. Завдяки такому підходу літні пацієнти зможуть відправляти фотографії тонометрів у звичні чати месенджерів без необхідності опановувати нові складні додатки, що робить розроблене алгоритмічне забезпечення максимально інклюзивним та ефективним інструментом для розвитку екосистеми електронного здоров'я. Сформовано далекоглядні перспективні напрями подальшого розвитку проєкту, серед яких пріоритетними є інтеграція повністю автономних оптимізованих нейромережових моделей для забезпечення розпізнавання в офлайн-режимі, додавання потужної підсистеми побудови інтерактивних графіків динаміки артеріального тиску, а також реалізація

протоколів захищеної хмарної синхронізації та функцій автоматичного експорту медичних звітів для професійного аналізу лікуючим лікарем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Артеріальна осцилографія: нові інформаційні можливості вимірювача артеріального тиску з програмним комплексом Оранта-АО : навч.-метод. посіб. , за ред. Д. В. Вакуленка, Л. О. Вакуленко. Львів : Магнолія, 2023. 508 с. URL: <https://magnolia.lviv.ua/?p=3108> (дата звернення: 01.05.2026).
2. Вакуленко Д. В., Вакуленко Л. О. Інформаційні технології у фізичній та реабілітаційній медицині : навч. посіб. Тернопіль : ТНПУ ім. В. Гнатюка, 2019. 205 с.
3. Дворцов Н. Дослідження та принципи побудови системи інтелектуального розпізнавання з використанням особливих точок. *Збірник праць молодих науковців ЦНТУ*. 2026. Вип. 16, ч. 1. С. 126–134. URL: <https://kntu.kr.ua/file/content/28037,n-dvortsov-doslidzhennia-ta-pryntsyypu-pobudovy-systemy-intelektualnoho-rozpiznavannia-z-vykorystanniam-osoblyvykh-tochok-str-126-134-.pdf> (дата звернення: 25.05.2026).
4. Мобільні технології для моніторингу показників стану здоров'я. *Вісник соціальної гігієни та організації охорони здоров'я України*. 2017. № 3. URL: <https://doi.org/10.11603/1681-2786.2017.3.8267> (дата звернення: 15.05.2026).
5. Федасюк Д., Цимбалюк Т., Руднєв В. Мобільний застосунок для перекладу та візуалізації тексту в доповненій реальності з використанням штучного інтелекту. *Information Systems and Networks*. 2025. Вип. 18, ч. 2. С. 71–90. URL: <https://doi.org/10.23939/sisn2025.18.2.071> (дата звернення: 16.05.2026).
6. Android RecyclerView documentation , Android Developers. URL: <https://developer.android.com/develop/ui/views/layout/recyclerview> (дата звернення: 29.05.2026).
7. App architecture guide , Android Developers. URL: <https://developer.android.com/topic/architecture> (дата звернення: 25.05.2026).
8. Blood pressure monitors user manuals and product documentation , Omron Healthcare. URL: <https://www.omron-healthcare.com>, (дата звернення: 24.05.2026).

9. Bradski G., Kaehler A. Learning OpenCV. Sebastopol : O'Reilly Media, 2008. 556 p.
10. CameraX overview , Android Developers. URL: <https://developer.android.com/training/camerax> (дата звернення: 27.05.2026).
11. Design guidelines and components , Material Design. URL: <https://m3.material.io>, (дата звернення: 18.05.2026).
12. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Ph.D. dissertation. Irvine : University of California, 2000. 162 p.
13. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 560 p.
14. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Reading : Addison-Wesley, 1994. 395 p.
15. Gemini API documentation , Google AI for Developers. URL: <https://ai.google.dev/gemini-api/docs> (дата звернення: 05.05.2026).
16. Gemini API generate content , Google AI for Developers. URL: <https://ai.google.dev/api/generate-content> (дата звернення: 06.05.2026).
17. Gonzalez R. C., Woods R. E. Digital Image Processing. 4th ed. New York : Pearson, 2017. 1192 p.
18. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. 800 p.
19. Google JSON documentation and org.json API references , Android Developers. URL: <https://developer.android.com/reference/org/json/package-summary> (дата звернення: 26.05.2026).
20. Guidelines and scientific documents on hypertension and cardiovascular prevention , European Society of Cardiology. URL: <https://www.escardio.org/Guidelines> (дата звернення: 11.05.2026).

21. Hypertension , World Health Organization. URL: <https://www.who.int/news-room/fact-sheets/detail/hypertension> (дата звернення: 20.05.2026).
22. ISO 9241-11:2018. Ergonomics of human-system interaction - Part 11: Usability: Definitions and concepts. Geneva : ISO, 2018. 29 p.
23. ISO,IEC 25010:2011. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. Geneva : ISO, 2011. 34 p.
24. Jetpack AppCompat documentation , Android Developers. URL: <https://developer.android.com/jetpack/androidx/releases/appcompat> (дата звернення: 07.05.2026).
25. Kotlin coroutines on Android , Android Developers. URL: <https://developer.android.com/kotlin/coroutines> (дата звернення: 25.05.2026).
26. Kotlin language documentation , Kotlin Documentation. URL: <https://kotlinlang.org/docs/home.html> (дата звернення: 22.05.2026).
27. Lifecycle-aware components , Android Developers. URL: <https://developer.android.com/topic/libraries/architecture/lifecycle> (дата звернення: 23.05.2026).
28. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 p.
29. Nielsen J. Usability Engineering. San Francisco : Morgan Kaufmann, 1993. 362 p.
30. Nixon M., Aguado A. Feature Extraction and Image Processing for Computer Vision. 4th ed. Oxford : Academic Press, 2019. 642 p.
31. Norman D. The Design of Everyday Things. Revised and expanded ed. New York : Basic Books, 2013. 368 p.
32. O'Gorman L., Sammon M. J., Seul M. Practical Algorithms for Image Analysis. 2nd ed. Cambridge : Cambridge University Press, 2008. 356 p.

33. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. New York : McGraw-Hill, 2019. 608 p.
34. Room persistence library , Android Developers. URL: <https://developer.android.com/training/data-storage/room> (дата звернення: 25.05.2026).
35. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Hoboken : Pearson, 2020. 1136 p.
36. Shneiderman B., Plaisant C. Designing the User Interface: Strategies for Effective Human-Computer Interaction. 6th ed. Boston : Pearson, 2016. 608 p.
37. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2015. 816 p.
38. SQLite database engine , SQLite Documentation. URL: <https://www.sqlite.org/docs.html> (дата звернення: 20.05.2026).
39. Szeliski R. Computer Vision: Algorithms and Applications. 2nd ed. Cham : Springer, 2022. 931 p.
40. Understanding blood pressure readings , American Heart Association. URL: <https://www.heart.org> (дата звернення: 20.05.2026).

ДОДАТКИ

Додаток А

Фрагменти програмного коду основних модулів

Головний клас *MainActivity*

Фрагмент показує ініціалізацію основних сервісів, підключення Room, створення агрегатора статистики та сервісу Gemini, а також запуск камери.

```
class MainActivity : AppCompatActivity() {
    companion object {
        private const val TAG = "GEMINI_OCR"
        private const val GEMINI_MODEL = "gemini-2.5-flash"
        private const val PREFS_NAME = "healthy_ranges"
    }

    private lateinit var measurementDao: MeasurementDao
    private lateinit var statsAiViewModel: StatsAiViewModel
    private lateinit var measurementAdapter: MeasurementAdapter
    private var healthyRanges = HealthyRanges(90, 119, 60, 79, 60, 100)

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        measurementDao = AppDatabase.getInstance(this).measurementDao()
        val measurementRepository = RoomMeasurementRepository(measurementDao)
        val metricsAggregator = MetricsAggregator(measurementRepository)
        val aiInsightsService = GeminiAiInsightsService(
            apiKey = BuildConfig.GEMINI_API_KEY.trim(),
            languageTagProvider = { currentAppLanguageTag() }
        )
        statsAiViewModel = ViewModelProvider(
            this,
            StatsAiViewModelFactory(metricsAggregator, aiInsightsService)
        ) [StatsAiViewModel::class.java]

        healthyRanges = loadHealthyRanges()
        bindViews()
        setupDatabaseObserver()
        setupListeners()
        observeAiState()
        showCameraPane()

        if (hasCameraPermission()) {
            startCamera()
        } else {
            cameraPermissionLauncher.launch(Manifest.permission.CAMERA)
        }
    }
}
```

Продовження додатку А

```
    }
}
```

Фрагмент OCR-обробки демонструє зняття кадру, попередню обробку зображення, виклик Gemini та збереження вимірювання.

```
private fun captureAndReadAndSaveMeasurement() {
    val capture = imageCapture ?: return
    setProcessingState(true)
    cameraSound.play(MediaActionSound.SHUTTER_CLICK)

    val imageFile = File.createTempFile("bp_capture_", ".jpg", cacheDir)
    val outputOptions = ImageCapture.OutputFileOptions.Builder(imageFile).build
    ()

    capture.takePicture(
        outputOptions,
        cameraExecutor,
        object : ImageCapture.OnImageSavedCallback {
            override fun onImageSaved(outputFileResults: ImageCapture.OutputFile
eResults) {
                val baseBitmap = BitmapFactory.decodeFile(imageFile.absolutePat
h) ?: return
                val rotated = rotateBitmapIfNeeded(baseBitmap, imageFile.abso
lutePath)

                val framedCrop = cropToOverlayFrame(rotated)
                val resizedCrop = shrinkBitmapMaxSide(framedCrop, 900)

                lifecycleScope.launch {
                    val ocrAttempt = withContext(Dispatchers.IO) {
                        recognizeWithGemini(resizedCrop)
                    }

                    if (ocrAttempt.result != null) {
                        withContext(Dispatchers.IO) {
                            measurementDao.insert(
                                Measurement(
                                    systolic = ocrAttempt.result.values [0],
                                    diastolic = ocrAttempt.result.values [1],
                                    pulse = ocrAttempt.result.values [2],
                                    timestamp = System.currentTimeMillis()
                                )
                            )
                        }
                    }

                    imageFile.delete()
                    setProcessingState(false)
                }
            }
        }
    )
}
```

Продовження додатку А

```

    )
}

```

Фрагмент розпізнавання показує, що застосунок передає в Gemini зображення у форматі JPEG та очікує JSON-відповідь із показниками.

```

private fun recognizeWithGemini(bitmap: Bitmap): OcrAttempt {
    if (geminiApiKey.isEmpty()) return OcrAttempt(null, "missing_api_key")

    val stream = ByteArrayOutputStream()
    bitmap.compress(Bitmap.CompressFormat.JPEG, 50, stream)
    val base64Image = Base64.encodeToString(stream.toByteArray(), Base64.NO_WRAP)

    val prompt = """
        This is Omron digital tonometer readings on LCD display.
        Read the values from the LCD and return JSON only in this exact format:
        {"systolic":123,"diastolic":88,"pulse":71,"confidence":0.93}
    """.trimIndent()

    val requestBody = JSONObject().apply {
        put(
            "contents",
            JSONArray().put(
                JSONObject().put(
                    "parts",
                    JSONArray()
                        .put(JSONObject().put("text", prompt))
                        .put(
                            JSONObject().put(
                                "inline_data",
                                JSONObject()
                                    .put("mime_type", "image/jpeg")
                                    .put("data", base64Image)
                            )
                        )
                )
            )
        )
    }.toString()

    val endpoint =
        "https://generativelanguage.googleapis.com/v1beta/models/$GEMINI_MODEL:generateContent?key=$geminiApiKey"
    val httpResult = postJson(endpoint, requestBody)
    val responseText = httpResult.body ?: return OcrAttempt(null, "api_request_failed")
    val content = extractGeminiText(responseText) ?: return OcrAttempt(null, "empty_model_content")
}

```

Продовження додатку А

```

    return OcrAttempt(parseModelResult(content).result, "parsed")
}

```

Сервіс *GeminiAiInsightsService*

Фрагмент показує генерацію текстових AI-висновків на основі агрегованих метрик.

```

class GeminiAiInsightsService(
    private val apiKey: String,
    private val modelName: String = "gemini-2.5-flash",
    private val promptBuilder: InsightsPromptBuilder = InsightsPromptBuilder(),
    private val languageTagProvider: () -> String = { Locale.getDefault().toLocaleTag() }
) : AiInsightsService {

    override suspend fun generateInsights(payload: MetricsPayload): String {
        val key = apiKey.trim()
        check(key.isNotEmpty()) { "Missing Gemini API key" }
        val languageTag = languageTagProvider().ifBlank { "en" }

        val requestBody = JSONObject().apply {
            put(
                "contents",
                JSONArray().put(
                    JSONObject().put(
                        "parts",
                        JSONArray().put(
                            JSONObject().put("text", promptBuilder.buildPrompt(
                                payload, languageTag))
                        )
                    )
                )
            )
            put("generationConfig", JSONObject().put("temperature", 0.2))
        }.toString()

        val endpoint = "https://generativelanguage.googleapis.com/v1beta/models/$modelName:generateContent?key=$key"
        val httpResult = postJson(endpoint, requestBody)
        val responseText = httpResult.body ?: throw IllegalStateException("AI request failed")
        return extractGeminiText(responseText).orEmpty().trim()
    }
}

```

Сутність *Measurement*

```

@Entity(tableName = "measurements")
data class Measurement(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,

```

Продовження додатку А

```

    val systolic: Int,
    val diastolic: Int,
    val pulse: Int,
    val timestamp: Long
)

```

DAO *MeasurementDao*

```

@Dao
interface MeasurementDao {
    @Insert
    suspend fun insert(measurement: Measurement)

    @Delete
    suspend fun delete(measurement: Measurement)

    @Query("SELECT * FROM measurements ORDER BY timestamp DESC")
    fun getAllSortedByTimestamp(): LiveData<List<Measurement>>

    @Query("SELECT * FROM measurements WHERE timestamp BETWEEN :startInclusive
AND :endInclusive ORDER BY timestamp ASC")
    suspend fun getMeasurementsBetween(startInclusive: Long, endInclusive: Long
): List<Measurement>

    @Query("DELETE FROM measurements")
    suspend fun deleteAll()
}

```

Агрегатор *MetricsAggregator*

```

class MetricsAggregator(
    private val repository: MeasurementRepository,
    private val periodBoundsCalculator: PeriodBoundsCalculator = PeriodBoundsCa
lculator(),
    private val zoneId: ZoneId = ZoneId.systemDefault()
) {
    suspend fun aggregateForLastCompletedPeriod(periodType: StatsPeriodType): A
ggregationResult? {
        val bounds = periodBoundsCalculator.lastCompletedPeriod(periodType)
        val measurements = repository.getMeasurementsBetween(bounds.startEpochM
illis, bounds.endEpochMillis)
        if (measurements.isEmpty()) return null

        val payload = MetricsPayload(
            periodType = periodType,
            periodStart = bounds.periodStart.format(DateTimeFormatter.ISO_LOCAL
_DATE),
            periodEnd = bounds.periodEnd.format(DateTimeFormatter.ISO_LOCAL_DAT
E),
            unit = "mmHg",
            metrics = AggregatedMetrics(

```

Продовження додатку А

```

        sbp = aggregateMetric(measurements.map { it.systolic }),
        dbp = aggregateMetric(measurements.map { it.diastolic }),
        pulse = aggregateMetric(measurements.map { it.pulse })
    ),
    counts = MeasurementCounts(
        measurements = measurements.size,
        daysWithMeasurements = measurements
            .map { Instant.ofEpochMilli(it.timestamp).atZone(zoneId).to
LocalDate() }
            .distinct()
            .size
    ),
    timeOfDay = aggregateTimeOfDay(measurements),
    notes = PayloadNotes(
        classificationImplemented = false,
        trendComputed = false,
        missingDataHandling = "No imputation; aggregates are computed o
nly from saved measurements."
    )
)

return AggregationResult(bounds = bounds, payload = payload)
}
}

```

Структура бази даних

Сутність *Measurement*

```

@Entity(tableName = "measurements")
data class Measurement(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val systolic: Int,
    val diastolic: Int,
    val pulse: Int,
    val timestamp: Long
)

```

Поле *id* використовується як первинний ключ із автоматичною генерацією. Числові поля *systolic*, *diastolic* і *pulse* зберігають показники тонометра, а *timestamp* фіксує момент створення запису в Unix-часі.

Клас бази даних *AppDatabase*

```

@Database(entities = [Measurement::class], version = 1, exportSchema = false)
abstract class AppDatabase : RoomDatabase() {
    abstract fun measurementDao(): MeasurementDao

    companion object {
        @Volatile
        private var INSTANCE: AppDatabase? = null

        fun getInstance(context: Context): AppDatabase {
            return INSTANCE ?: synchronized(this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    AppDatabase::class.java,
                    "measurements_db"
                ).build()
                INSTANCE = instance
                instance
            }
        }
    }
}

```

База даних містить одну сутність *Measurement* і надає доступ до DAO через метод *measurementDao()*. Ім'я локальної бази даних - *measurements_db*.

DAO *MeasurementDao*

```

@Dao
interface MeasurementDao {
    @Insert
    suspend fun insert(measurement: Measurement)
}

```

Продовження додатку Б

```

@Delete
suspend fun delete(measurement: Measurement)

@Query("SELECT * FROM measurements ORDER BY timestamp DESC")
fun getAllSortedByTimestamp(): LiveData<List<Measurement>>

@Query("SELECT * FROM measurements WHERE timestamp BETWEEN :startInclusive
AND :endInclusive ORDER BY timestamp ASC")
suspend fun getMeasurementsBetween(startInclusive: Long, endInclusive: Long
): List<Measurement>

@Query("DELETE FROM measurements")
suspend fun deleteAll()
}

```

DAO підтримує основні операції над таблицею: вставлення нового запису, видалення одного запису, отримання списку для екрану історії, вибірку за періодом та повне очищення таблиці.

Логічна структура таблиці

Поле	Тип	Призначення
<i>id</i>	<i>Int</i>	Первинний ключ, автоінкремент
<i>systolic</i>	<i>Int</i>	Систолічний тиск
<i>diastolic</i>	<i>Int</i>	Діастолічний тиск
<i>pulse</i>	<i>Int</i>	Пульс
<i>timestamp</i>	<i>Long</i>	Час вимірювання