Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

До захисту допущено:
В.о. завідувача кафедри

Артем ВОЛОКИТА

2026 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»
спеціальність 121 — «Інженерія програмного забезпечення»

МОБІЛЬНИЙ ЗАСТОСУНОК МОНІТОРИНГУ СЕРЦЕВОГО РИТМУ НА ОСНОВІ АНАЛІЗУ ВІДЕОПОТОКУ З КАМЕРИ ПРИСТРОЮ

Виконала
Дворецька Анастасія Вадимівна, студентка 4 курсу, група ІМ-21

Керівник
Трочун Євгеній Володимирович, д-р. філос., асистент

Консультант
Міщенко Людмила Дмитрівна, ст. вик., док. філ. комп. інж.

Рецензент
Шимкович Володимир Миколайович, кандидат технічних наук, доцент

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

Дворецька А. В.

Київ — 2026 р.

АНОТАЦІЯ

Актуальність. Серцево-судинні захворювання залишаються провідною причиною смертності, тому регулярний контроль ЧСС важливий для самоспостереження. Класичні методи (ЕКГ, пульсоксиметрія) потребують окремого обладнання, тоді як камера сучасного смартфона дозволяє вимірювати пульс методом фотоплетизмографії (PPG) без додаткових пристроїв.

Мета роботи — проєктування та програмна реалізація мобільного застосунку для Android, що обчислює ЧСС за відеопотоком з камери методом PPG.

Для досягнення мети вирішено такі **задачі**: проаналізувати наявні методи вимірювання ЧСС та застосунки-аналоги; обґрунтувати вибір PPG-методу та сформулювати вимоги до архітектури; спроєктувати застосунок за принципами Clean Architecture з шарами presentation, domain, data; реалізувати захоплення відеопотоку, виділення ROI, цифрову обробку сигналу та обчислення ЧСС; реалізувати збереження історії вимірювань та інтерфейс користувача; перевірити працездатність на реальному пристрої.

Особистий вклад. Самостійно спроєктовано архітектуру застосунку, обрано стек (Kotlin, CameraX, Coroutines, Room), реалізовано модулі захоплення кадрів, виявлення пальця, цифрової обробки сигналу (видалення тренду базової лінії, смугова фільтрація, виявлення піків) та обчислення BPM. Розроблено інтерфейс і базу історії вимірювань. Проведено пілотну перевірку точності.

Ключові слова: *частота серцевих скорочень, фотоплетизмографія, PPG, мобільний застосунок, Android, Kotlin, обробка сигналів.*

ANNOTATION

The aim of this work is to design and implement an Android mobile application that measures the user's heart rate from the smartphone camera video stream using the photoplethysmography (PPG) method.

To reach this aim the following objectives were solved: analyse existing HR measurement methods and competing applications; justify the choice of PPG and formulate architecture requirements; design the application following Clean Architecture with presentation, domain and data layers; implement video capture, region-of-interest extraction, digital signal processing and HR estimation; implement measurement history storage and the user interface; verify the application on a real device.

Keywords: *heart rate, photoplethysmography, PPG, mobile application, Android, Kotlin, signal processing.*

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень: Бакалавр
Освітньо-професійна програма: «Інженерія програмного забезпечення комп'ютерних систем»
Спеціальність: 121 — «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри

Артем ВОЛОКИТА

2026 р.

ЗАВДАННЯ
на бакалаврський дипломний проєкт

Дворецькій Анастасії Вадимівні

1. Тема проєкту: Мобільний застосунок моніторингу серцевого ритму на основі аналізу відеопотоку з камери пристрою
   керівник Трочун Євгеній Володимирович, д-р. філос., асистент
   затверджено наказом по університету від 3 червня 2026 року №2055-с

2. Дата здачі закінченого проєкту 3 червня 2026 року

3. Вихідні дані для проекту: технічна документація, теоретичні та статистичні дані, патенти на винахід, наукові публікації.

4. Зміст розрахунково-пояснювальної записки: Огляд існуючих рішень у задачі моніторингу серцевого ритму. Огляд технологій розробки мобільного застосунку. Деталі проєктування та програмної реалізації системи. Дослідження та аналіз розробленого застосунку.

5. Перелік графічного матеріалу: Принципова схема. Функціональна схема. Структурна схема. Лістинг коду.

6. Консультанти розділів проєкту

№	Розділ	Консультант (прізвище та ініціали)	Підпис, дата	
			завдання видав	завдання прийняв
1	Нормоконтроль	Міщенко Л. Д.		

7. Дата видачі завдання 9 лютого 2026 року

Календарний план

№	Назва етапу виконання дипломного проекту	Термін виконання	Примітка
1	Затвердження теми проєкту	09.02.2026–03.04.2026	
2	Вивчення та аналіз завдання	24.03.2026–20.04.2026	
3	Розробка архітектури та загальної структури системи	22.04.2026–17.05.2026	
4	Програмна реалізація системи	17.05.2026–31.05.2026	
5	Оформлення пояснювальної записки	10.05.2026–02.06.2026	
6	Передзахист	03.06.2026	
7	Захист	16.06.2026	

Виконавець Дворецька Анастасія Вадимівна

Керівник Трочун Євгеній Володимирович

Довідки	Формат	Позначення	Найменування	Кільк. листів	№ екземпляра	Додаток
	А4		Завдання на дипломний проект	2		
	А4	ІАЛЦ.467200.002 ТЗ	Моніторинг серцевого ритму за	1		
			відеопотоком камери смартфона			
			Технічне завдання			
	А4	ІАЛЦ.467200.003 ПЗ	Моніторинг серцевого ритму за	74		
			відеопотоком камери смартфона			
			Пояснювальна записка			
	А4	ІАЛЦ.467200.004 ДА	Принципова схема	1		
	А4	ІАЛЦ.467200.005 ДБ	Функціональна схема	1		
	А4	ІАЛЦ.467200.006 ДВ	Структурна схема	1		
	А4	ІАЛЦ.467200.007 ДГ	Лістинг коду	4		

Зм.	Лист	№ докум.	Підп.	Дата	ІАЛЦ.467200.001 ОА		
Розробив		Дворецька А. В.			Моніторинг серцевого ритму за відеопотоком камери смартфона Опис альбому	Лит.	Аркуш 1 / Аркушів 1
Перевірив		Трочун Є. В.					
Н. контр.		Міщенко Л. Д.				НТУУ "КПІ ім. Ігоря Сікорського", ФІОТ ІМ-21	
Затвердив		Волокита А. М.					

МОНІТОРИНГ СЕРЦЕВОГО РИТМУ ЗА ВІДЕОПОТОКОМ КАМЕРИ СМАРТФОНА

Технічне завдання

ІА/ЛЦ.467200.002 ТЗ

Київ – 2026 р.

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання полягає у розробці мобільного застосунку моніторингу серцевого ритму на основі аналізу відеопотоку з камери пристрою, а також на його подальшу підтримку та вдосконалення.

Областю застосування системи є побутове використання у контексті оздоровчого фітнес моніторингу: самоспостереження за пульсом у стані спокою, контроль ЧСС до та після фізичних навантажень, ведення особистої історії вимірювань. Застосунок не є медичним виробом і не призначений для встановлення діагнозу або заміни консультації кваліфікованого медичного працівника.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання для виконання дипломного проєкту бакалаврського рівня вищої освіти, що було затверджене кафедрою обчислювальної техніки факультету інформатики та обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням розробки є створення мобільного застосунку, що забезпечує вимірювання частоти серцевих скорочень користувача за відеопотоком, отриманим із вбудованої камери смартфона, методом фотоплетизмографії. Застосунок надає користувачеві засіб для оперативного контролю стану серцево-судинної системи без потреби у додатковому обладнанні та накопичує історію раніше виконаних вимірювань для відстеження динаміки показника.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки даного дипломного проєкту є офіційна документація платформи Android та бібліотек, що використовуються (CameraX, Room, Kotlin Coroutines), наукові публікації з фотоплетизмографії та цифрової обробки сигналу, технічна та довідкова література, відкриті ресурси мережі Інтернет за тематикою роботи.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

- Захоплення відеопотоку із задньої камери мобільного пристрою з автоматичним вмиканням ліхтарика на період вимірювання.
- Автоматичне визначення моменту прикладення пальця до об’єктива камери за рівнем яскравості кадру і перехід у режим вимірювання.
- Виділення червоного колірного каналу у центрі кадру для формування PPG-сигналу.
- Цифрова обробка отриманого сигналу: видалення тренду базової лінії та смугова фільтрація у діапазоні 0,7–3,5 Гц.
- Детекція піків PPG-сигналу та обчислення ЧСС на основі аналізу міжударних інтервалів.
- Відображення форми сигналу у реальному часі на екрані під час вимірювання.
- Збереження результатів вимірювань (значення ЧСС, дата, час) у локальній базі даних пристрою.
- Перегляд історії раніше виконаних вимірювань та її очищення за запитом користувача.

– Інтуїтивно зрозумілий інтерфейс з підказками щодо правильного розміщення пальця на камері.

5.2. Вимоги до програмного забезпечення

– Цільова операційна система — Android версії 8.0 (API 26) або вище.

5.3. Вимоги до апаратної частини

– Задня камера мобільного пристрою з підтримкою автофокусу, роздільною здатністю не нижче 2 Мп і наявним LED-спалахом.
– Оперативна пам’ять — не менше 2 ГБ.
– Вільне місце у внутрішній пам’яті — не менше 100 МБ для встановлення застосунку та зберігання історії вимірювань.

6. ЕТАПИ РОЗРОБКИ

Назва етапу виконання	**Термін виконання**
Затвердження теми проєкту	09.02.2026–03.04.2026
Вивчення та аналіз завдання	24.03.2026–20.04.2026
Розробка архітектури та загальної структури системи	22.04.2026–17.05.2026
Програмна реалізація системи	17.05.2026–27.05.2026
Тестування та виправлення помилок	27.05.2026–31.05.2026
Оформлення пояснювальної записки	10.05.2026–02.06.2026

МОНІТОРИНГ СЕРЦЕВОГО РИТМУ ЗА ВІДЕОПОТОКОМ КАМЕРИ СМАРТФОНА

Пояснювальна записка

ІА/ЛЦ.467200.003 ПЗ

Київ – 2026 р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ВООЗ	Всесвітня організація охорони здоров’я
ЕКГ	Електрокардіографія, електрокардіограма
ССЗ	Серцево-судинні захворювання
ЧСС	Частота серцевих скорочень
API	(Application Programming Interface) інтерфейс програмування застосунків
APK	(Android Package) формат інсталяційного пакета для Android
BPM	(Beats Per Minute) ударів за хвилину
CMOS	(Complementary Metal-Oxide-Semiconductor) комплементарний метало-оксидний напівпровідник
DAO	(Data Access Object) об’єкт доступу до даних
DTO	(Data Transfer Object) об’єкт передачі даних
FFT	(Fast Fourier Transform) швидке перетворення Фур’є
FIR	(Finite Impulse Response) фільтр зі скінченною імпульсною характеристикою
HRV	(Heart Rate Variability) варіабельність серцевого ритму
IBI	(Inter-Beat Intervals) міжударні інтервали
ICA	(Independent Component Analysis) аналіз незалежних компонент
IDE	(Integrated Development Environment) інтегроване середовище розробки
IIR	(Infinite Impulse Response) фільтр з нескінченною імпульсною характеристикою

JSON	(JavaScript Object Notation) формат обміну даними на основі JavaScript
LED	(Light-Emitting Diode) світловипромінювальний діод
OS	(Operating System) операційна система
PPG	(Photoplethysmography) фотоплетизмографія
RGB	(Red, Green, Blue) колірна модель з червоним, зеленим і синім каналами
ROI	(Region of Interest) область інтересу
rPPG	(remote Photoplethysmography) безконтактна фотоплетизмографія
SDK	(Software Development Kit) набір засобів розробки
SpO2	(Peripheral Oxygen Saturation) сатурація крові киснем
SQL	(Structured Query Language) мова структурованих запитів
UI	(User Interface) інтерфейс користувача
UX	(User Experience) користувацький досвід
XML	(eXtensible Markup Language) розширювана мова розмітки
YUV	(Luma, Chroma) колірна модель з компонентами яскравості та кольоровості

ВСТУП

Стан серцево-судинної системи людини традиційно вважається одним із ключових індикаторів загального здоров'я, а частота серцевих скорочень (ЧСС) серед усіх фізіологічних показників вирізняється тим, що її можна виміряти у будь-якій ситуації — удома, у спортивному залі, перед сном. За даними Всесвітньої організації охорони здоров'я, серцево-судинні захворювання щорічно забирають близько 17,9 млн життів і залишаються головною причиною смертності у світі. Регулярне відстеження пульсу у спокої, під час навантажень та після них дозволяє своєчасно зауважити порушення серцевого ритму ще до моменту, коли вони переходять у форму, що потребує медичного втручання.

Існує чимало інструментів, які вимірюють ЧСС досить точно: електрокардіографи, медичні пульсоксиметри, носимі фітнес-трекери та смарт-годинники. Спільна проблема цих засобів — вони або потребують спеціального обладнання, або коштують відчутних грошей, або просто рідко опиняються поряд у момент, коли вимірювання було б корисним. На противагу їм смартфон у переважної більшості користувачів завжди при собі, а його технічні характеристики останніх років цілком достатні для того, щоб роль оптичного датчика пульсу виконувала вбудована камера. У цьому й полягає принципова ідея методу фотоплетизмографії (PPG): при притисканні кінчика пальця до об'єктива задньої камери разом із увімкненим ліхтариком тканини періодично змінюють свою прозорість синхронно з кожним серцевим скороченням, що відбивається у послідовності кадрів як ритмічне коливання яскравості. Залишається лише коректно виокремити це коливання з відеопотоку та обчислити його частоту.

Цей метод не є новим, він описаний у науковій літературі ще з 1930-х років, але його реалізація саме на мобільних пристроях стала можливою

лише у 2010-х, коли продуктивність процесорів і якість камер досягли необхідного рівня. У магазині застосунків Google Play представлено вже не один десяток рішень, які заявляють про можливість вимірювати пульс пальцем на камері. Втім, аналіз цих рішень показує, що значна частина з них використовує спрощені алгоритми обробки відеосигналу, не забезпечує стабільної точності між сеансами і не дає користувачу зрозумілого уявлення про якість отриманого вимірювання.

У межах дипломного проєкту виконано розробку мобільного застосунку для платформи Android, який реалізує описаний підхід із застосуванням покращеного алгоритму обчислення ЧСС. Покращення стосуються трьох ключових аспектів. По-перше, для формування первинного сигналу використовується не усереднена яскравість усього кадру, а виключно червоний колірний канал у централізованій області інтересу — адже саме він найбільш чутливий до коливань кровонаповнення капілярів. По-друге, до сигналу застосовується смугова цифрова фільтрація у діапазоні 0,7–3,5 Гц, що відповідає фізіологічно можливим значенням ЧСС від 42 до 210 ударів за хвилину, разом із попереднім видаленням повільного тренду базової лінії. По-третє, для обчислення BPM використовується аналіз міжударних інтервалів між виявленими піками сигналу замість прямого підрахунку локальних максимумів за фіксований проміжок часу. Реалізацію виконано на мові Kotlin із застосуванням бібліотек CameraX для роботи з відеопотоком, Kotlin Coroutines та Flow для асинхронної обробки даних, Room для збереження історії вимірювань; архітектура застосунку базується на принципах багатошарової архітектури з виділеними шарами presentation, domain та data.

Метою дипломного проєкту є розробка мобільного застосунку для платформи Android, призначеного для вимірювання частоти серцевих скорочень користувача методом фотоплетизмографії з опрацюванням відеопотоку з вбудованої камери пристрою. Об’єктом дослідження виступає процес неін-

вазивного моніторингу серцевого ритму в умовах побутового застосування, а предметом — методи обчислення ЧСС за послідовністю відеокадрів та технології мобільної розробки, що забезпечують їх ефективну реалізацію. Для досягнення мети у роботі проаналізовано існуючі підходи до моніторингу серцевого ритму, виявлено сильні та слабкі сторони наявних на ринку мобільних застосунків подібного призначення, обґрунтовано вибір технологічного стеку, спроєктовано архітектуру та алгоритм обробки сигналу, виконано програмну реалізацію та проведено її тестування. Розроблений застосунок орієнтований на побутове використання у контексті оздоровчо-фітнес моніторингу і не позиціонується як медичний виріб.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Контекст задачі моніторингу частоти серцевих скорочень

Серце є органом, який працює без перерви протягом усього життя людини, і навіть незначні зміни у режимі його роботи здатні сигналізувати про порушення у роботі інших систем організму. Частота серцевих скорочень — кількість ударів серця за одиницю часу — це найбільш базовий і водночас найбільш інформативний показник, який можна виміряти без спеціальних умов. Нормальним діапазоном ЧСС у стані спокою для дорослої здорової людини прийнято вважати інтервал від 60 до 100 ударів за хвилину [1]; систематичне відхилення від цього діапазону може вказувати на тахікардію, брадикардію, аритмію та інші стани, що потребують уваги.

Можливість регулярно вимірювати ЧСС у різних побутових ситуаціях важлива не лише для людей із діагностованими захворюваннями серцево-судинної системи, але й для широкого кола користувачів, яким цікаво стежити за реакцією організму на фізичні навантаження, оцінювати ефективність тренувань, контролювати рівень стресу або просто формувати загальне уявлення про власний стан здоров’я. За даними дослідницьких звітів останніх років, попит на технології персонального моніторингу здоров’я постійно зростає, що підтверджується як збільшенням кількості користувачів носимих пристроїв, так і розширенням функціональності мобільних застосунків відповідної тематики.

Класичний підхід до вимірювання ЧСС передбачав застосування медичних приладів у клінічних умовах. З розвитком електронної компонентної бази з’явилися компактні носимі пристрої — спершу нагрудні кардіомоніто-

ри для спортсменів, потім фітнес-браслети та смарт-годинники. Втім, ці пристрої досі залишаються окремими гаджетами, які користувач має придбати, заряджати і носити при собі. Альтернативний шлях полягає у використанні апаратних можливостей смартфона, який вже є практично у кожної людини, без необхідності докуповувати додаткове обладнання. Саме цей шлях і досліджується у даній роботі.

1.2 Огляд методів вимірювання серцевого ритму

Існує кілька принципово різних способів виміряти частоту серцевих скорочень людини. Кожен із них має свої сильні та слабкі сторони, а вибір конкретного методу визначається призначенням пристрою, вимогами до точності, бюджетом проєкту та умовами експлуатації. Розглянемо основні з них.

1.2.1 Електрокардіографія

Електрокардіографія (ЕКГ) є золотим стандартом діагностики у кардіології. Метод базується на реєстрації електричних потенціалів, що виникають при поляризації та деполяризації м’язових клітин серця. Електроди розташовуються на поверхні тіла пацієнта у певних точках, з’єднуються з електрокардіографом, і отриманий запис ритмічної електричної активності серця — електрокардіограма — аналізується лікарем.

ЕКГ дозволяє не лише визначити ЧСС, але й виявити патології провідної системи серця, ішемічні зміни, аритмії та інші порушення [2]. Точність методу винятково висока, однак для класичної 12-канальної ЕКГ потрібне спеціальне обладнання, кваліфікований персонал та умови медичного закладу. У побутових умовах метод важко застосувати; винятком є портативні одноканальні ЕКГ-пристрої, які з’явилися в останні роки, але вони все ще становлять окрему категорію приладів, а не функцію смартфона.

1.2.2 Пульсова оксиметрія

Пульсова оксиметрія (SpO2) первинно призначена для вимірювання ступеня насичення гемоглобіну киснем. Принцип методу полягає у просвічуванні тонкої ділянки тканини (як правило, пальця або мочки вуха) двома світловими променями різної довжини хвилі (червоний близько 660 нм та інфрачервоний близько 940 нм) та вимірюванні інтенсивності світла, що пройшло крізь тканину. Оскільки оксигенований та дезоксигенований гемоглобін мають різні спектри поглинання, за співвідношенням сигналів двох каналів обчислюється рівень насичення киснем.

Як побічний результат пульсоксиметр також обчислює ЧСС: ритмічні коливання інтенсивності прийнятого світла, спричинені пульсуючим кровотоком в артеріях, дозволяють виявити моменти серцевих скорочень. Похибка вимірювання пульсу медичним пульсоксиметром становить порядку ±2–3 уд/хв, що є цілком прийнятним для більшості клінічних та побутових застосувань. Втім, як і у випадку з ЕКГ, медичний пульсоксиметр — це окремий пристрій, який не завжди є доступним.

1.2.3 Фотоплетизмографія

Фотоплетизмографія (англ. *photoplethysmography, PPG*) є безпосереднім родичем пульсоксиметрії, але як самостійний метод зосереджується саме на формі та частоті пульсової хвилі, не вимагаючи двох довжин хвиль для розрахунку SpO2. Принцип такий самий — просвічування тканини, реєстрація змінної складової світлового потоку, періодична модуляція якого синхронізована з серцевими скороченнями. Зміни оптичних властивостей тканин при цьому пов’язані із циклічним наповненням та спорожненням капілярів кров’ю.

Класична PPG, як було показано ще у роботах Hertzman у 1930-х ро-

ках та докладно описано у сучасних оглядах [2; 3], може бути реалізована як у трансмісійному режимі (приймач світла розташований по інший бік тканини відносно джерела), так і у рефлекторному (приймач знаходиться з того ж боку, що й джерело, і реєструє відбите світло). Останній варіант саме і використовується у носимих пристроях типу фітнес-браслетів та смарт-годинників, де на тильному боці корпусу розміщено зелений світлодіод та фотодіод, притиснуті до шкіри зап'ястка.

1.2.4 Безконтактна фотоплетизмографія

Окремим напрямом, який активно розвивається з кінця 2000-х років, є безконтактна (англ. *remote*) фотоплетизмографія — rPPG. У роботі Verkruysse, Svaasand та Nelson [4] було показано, що PPG-сигнал можна виокремити навіть із відеозапису обличчя людини за умов природного освітлення, без жодного фізичного контакту з пристроєм. Згодом Poh, McDuff та Picard у праці [5] запропонували метод сліпого розділення джерел сигналу (ICA), що суттєво підвищив якість виокремлення корисної складової. Окремий клас методів, заснований на колірних просторах, був представлений у роботі de Haan та Jeanne [6].

Безконтактна PPG є привабливою з точки зору зручності, оскільки взагалі не потребує жодних дій від користувача окрім перебування перед камерою; втім, її точність помітно поступається контактному варіанту і сильно залежить від умов освітлення, відстані до камери, руху голови, тону шкіри та інших факторів. Тому на сучасному рівні розвитку технологій rPPG використовується переважно у наукових дослідженнях та експериментальних розробках, тоді як для побутових застосунків зручнішим залишається контактний варіант, який і обрано у цій роботі.

1.2.5 Порівняльна характеристика методів

Зведене порівняння розглянутих методів за основними критеріями наведене у таблиці 1.1.

Таблиця 1.1 — Порівняння методів вимірювання ЧСС

Метод	**Точність**	**Обладнання**	**Доступність**	**Реалізація на смартфоні**
ЕКГ	дуже висока	висока	низька	неможлива
SpO_2	висока	середня	обмежена	потребує датчика
Контактна PPG	висока	низька	висока	камера + ліхтарик
Безконтактна rPPG	середня	дуже низька	висока	чутлива до умов

З наведеного аналізу видно, що для реалізації побутового мобільного застосунку оптимальним є саме контактний метод фотоплетизмографії з використанням камери смартфона: він поєднує прийнятну точність із відсутністю вимог до додаткового обладнання та достатньо стійкий до умов експлуатації за рахунок використання вбудованого LED-ліхтарика як джерела стабільного освітлення.

1.3 Принцип роботи фотоплетизмографії на камері смартфона

Розглянемо детальніше фізичні та технічні аспекти роботи методу, покладеного в основу розроблюваного застосунку.

1.3.1 Фізична природа сигналу

Шкіра людини, м'які тканини та капілярна сітка, що пронизує їх, мають певні оптичні властивості: поглинають, відбивають і розсіюють світло. При кожному скороченні серця через артеріоли та капіляри проходить пульсова хвиля — короткочасне підвищення тиску і відповідне підвищення кровонаповнення. Кров поглинає світло сильніше, ніж навколишні тканини (особливо у червоному діапазоні спектру), тому при притисканні пальця до камери з увімкненим спалахом фотосенсор камери реєструє ритмічне зменшення інтенсивності світла у моменти серцевого скорочення та її відновлення у періоди між скороченнями.

У формальному вигляді цей процес можна описати моделлю Beer-Lambert, згідно з якою інтенсивність світла $I(t)$, що дійшло до сенсора, пов'язана з товщиною шару поглинаючої речовини $d(t)$ співвідношенням:

$$I(t) = I_0 \cdot \exp\left(-\varepsilon \cdot c \cdot d(t)\right), \tag{1}$$

де I_0 — інтенсивність вихідного світла від джерела (ліхтарика), ε — молярний коефіцієнт поглинання речовини на відповідній довжині хвилі, c — молярна концентрація поглинача (гемоглобіну в крові), $d(t)$ — ефективна оптична товщина тканини у момент часу t, яка циклічно змінюється під впливом пульсової хвилі. Невелика змінна складова $\Delta d(t)$ при кожному серцевому скороченні і породжує спостережувані коливання інтенсивності, які в часі утворюють PPG-сигнал.

1.3.2 Особливості реєстрації сигналу на камері

Сучасні смартфонні камери побудовані за принципом матриці світлочутливих елементів CMOS, оснащеної масивом світлофільтрів Байєра. Це означає, що кожен піксель кінцевого кадру несе в собі інформацію про інтенсивність світла у одному з трьох колірних каналів — червоному (R), зеленому (G) або синьому (B). Завдяки особливостям спектрального поглинання гемоглобіну та глибині проникнення світла відповідних довжин хвиль у тканини, найбільш інформативним для PPG-сигналу при просвічуванні пальця виявляється червоний канал. Зелений канал також несе корисну інформацію та часто використовується у rPPG-системах, які працюють із відбитим світлом на зап'ястку чи обличчі; для контактної реєстрації пальцем оптимальним є саме червоний канал.

Просторовий розподіл сигналу по кадру також не є рівномірним. По периметру кадру можуть спостерігатися ефекти затемнення (вігнетування), нерівномірного підсвічування або просочування зовнішнього світла з-під пальця. З цієї причини при побудові сигналу доцільно усереднювати значення пікселів не за всім кадром, а лише за центральною областю інтересу — так званим ROI (*Region of Interest*). Розмір ROI обирається як компроміс між якістю усереднення (більше пікселів — менший шум) і стійкістю до периферійних артефактів (менший розмір — менше зайвих факторів). У типових реалізаціях ROI становить квадрат із стороною від 1/4 до 1/2 розміру кадру в центрі.

Окрім просторового вибору каналу та області, важливу роль відіграє постійне освітлення зони реєстрації. При вимкненому ліхтарику сигнал PPG може майже не виокремлюватися з шуму через дуже малу інтенсивність світла, що дійшло до сенсора крізь тканину пальця. Увімкнення режиму `FLASH_MODE_TORCH` камери забезпечує стабільне підсвічування і суттєво під-

вищує співвідношення сигнал-шум.

1.3.3 Етапи обробки відеосигналу

Загальна послідовність обробки PPG-сигналу, отриманого з камери смартфона, складається з кількох послідовних кроків. На першому етапі з відеопотоку камери послідовно отримуються кадри. Із кожного кадру виділяється центральна область інтересу і ізолюється червоний колірний канал, після чого усереднюється значення інтенсивності пікселів у межах ROI — так формується одне значення сирого сигналу на кадр.

Послідовність таких значень формує первинний PPG-сигнал. Він неминуче містить як корисну періодичну складову, так і небажані артефакти — повільний дрейф базової лінії (через зміну тиску пальця на камеру, тепловий дрейф сенсора тощо), високочастотний шум зчитування, а також можливі сплески від руху руки користувача. Для очищення сигналу застосовується послідовність цифрових фільтрів: спершу видаляється тренд базової лінії (наприклад, через відняття ковзного середнього або поліноміальну апроксимацію [7]), потім сигнал пропускається через смуговий фільтр з частотами зрізу, що відповідають фізіологічно можливим значенням ЧСС, тобто приблизно 0,7–3,5 Гц (що дорівнює 42–210 уд/хв).

На очищеному сигналі виконується пошук локальних максимумів — піків, які відповідають моментам систоли (скорочення лівого шлуночка). За часовою відстанню між сусідніми піками обчислюються міжударні інтервали (IBI, *Inter-Beat Intervals*), а середнє значення цих інтервалів використовується для розрахунку ЧСС за формулою:

$$\text{BPM} = \frac{60}{\overline{T}_{\text{IBI}}}, \tag{2}$$

де $\overline{T}_{\text{IBI}}$ — середній міжударний інтервал у секундах. Такий підхід є точнішим за пряме рахування кількості піків за фіксований проміжок часу,

оскільки нечутливий до того, у який момент починається і закінчується вікно вимірювання, і дозволяє коректно усереднити значення навіть за коротких записів тривалістю 5–10 с.

1.3.4 Джерела похибок та обмеження методу

Як і будь-який оптичний метод, фотоплетизмографія через камеру смартфона має низку природних обмежень, які впливають на точність вимірювання. Перерахуємо найважливіші з них:

- **Артефакти руху** — найвагоміший фактор спотворення. Будь-який рух пальця відносно об'єктива камери призводить до різкої зміни рівня сигналу, який легко прийняти за серцеве скорочення. Боротися з рухом можна як рекомендаціями користувачу тримати руку нерухомо, так і програмними методами — виявленням і вилученням «зіпсованих» ділянок запису.
- **Тиск пальця на камеру** — надмірний тиск перетискає капіляри і знижує амплітуду пульсової хвилі; надто слабкий тиск, навпаки, призводить до просочування зовнішнього світла. Оптимальний рівень тиску визначається емпірично і користувач часто потребує підказки.
- **Індивідуальні особливості тканини** — товщина шкіри, насиченість пігментом, температура поверхні впливають на абсолютний рівень сигналу. Втім, оскільки система працює з відносними коливаннями, ці відмінності рідко є критичними.
- **Низький рівень освітлення** — при вимкненому ліхтарику чи поганому контакті з камерою сигнал може просто не сформуватися. Обов'язковою умовою методу є наявність LED-спалаху і його коректне увімкнення на час вимірювання.
- **Аритмія та інші порушення серцевого ритму** — при значній не-

регулярності серцебиття стандартні алгоритми розрахунку усередненого IBI можуть давати помилкові результати. Для таких випадків потрібна окрема обробка, яка виходить за межі побутового використання.

1.4 Огляд існуючих мобільних застосунків

На сучасному ринку мобільних застосунків представлено значну кількість продуктів, які заявляють про вимірювання ЧСС за відеопотоком камери. Розглянемо найпопулярніших представників цього сегменту, проаналізуємо їх функціональні можливості, виявимо переваги та недоліки кожного.

1.4.1 Instant Heart Rate (Azumio)

Instant Heart Rate розробки компанії Azumio належить до найстаріших і найвідоміших застосунків такого типу. Продукт активно поширюється на платформах Android та iOS, у магазині Google Play має понад 50 млн встановлень [8]. Зовнішній вигляд застосунку лаконічний: користувачеві пропонується покласти палець на камеру і чекати, поки на екрані з’явиться отримане значення пульсу.

Серед переваг застосунку: проста і зрозуміла інструкція для першого використання, наявність історії вимірювань, можливість експорту даних та інтеграція зі сторонніми сервісами для відстеження здоров’я. Графіка інтерфейсу досить мінімалістична, що позитивно впливає на швидкодію навіть на бюджетних пристроях.

Серед недоліків можна назвати: чимало функцій, включно з розширеним аналізом і відсутністю реклами, доступні лише у платній підписці; інструкції щодо коректного розміщення пальця обмежені одним екраном і не дають жодного зворотного зв’язку у випадку, якщо палець перекриває камеру некоректно; формат PPG-сигналу під час вимірювання користувачеві не

показується.

1.4.2 Heart Rate Monitor (Runtastic)

Heart Rate Monitor від компанії Runtastic (тепер частина групи adidas) орієнтований передусім на спортивну аудиторію [9]. Застосунок інтегрується у ширшу екосистему фітнес-продуктів та дозволяє прив'язувати вимірювання до тренувальних сеансів. Інтерфейс має сучасний вигляд із характерними для бренду кольорами.

Переваги застосунку: гарна інтеграція з рештою сервісів Runtastic, наявність контекстуальних рекомендацій (наприклад, оцінка зони пульсу під час тренувань), статистика за різні періоди.

Недоліки: значна частина функцій вимагає створення облікового запису у сервісі та працює лише при наявності постійного інтернет-з'єднання; під час перших вимірювань спостерігається певна затримка перед видачею результату; саме обчислення пульсу відбувається у фіксованому вікні без можливості його налаштування; використовується спрощений алгоритм без явної смугової фільтрації сигналу.

1.4.3 Welltory

Welltory [10] позиціонує себе як комплексний застосунок для відстеження «розумного здоров'я», поєднуючи вимірювання ЧСС із функціями оцінки стресу та варіабельності серцевого ритму (HRV). Тривалість одного вимірювання у Welltory зазвичай більша, ніж у конкурентів: до 2 хвилин на запис, що пояснюється необхідністю накопичити статистично надійний обсяг даних для розрахунку HRV.

Переваги: глибокий аналіз отриманого сигналу, інтеграція з даними смарт-годинників, відображення тренду показників у часі.

Недоліки: довгий процес вимірювання може втомлювати користувача,

особливо якщо потрібен швидкий результат; основна частина функціональності закрита у платній підписці; інтерфейс перевантажений елементами, що може ускладнити використання людям без досвіду роботи з подібними продуктами.

1.4.4 Cardiio

Cardiio [11] вирізняється тим, що пропонує два режими вимірювання: контактний (палець на камері) та безконтактний (обличчя перед камерою). Безконтактний режим використовує технологію rPPG і працює за умов достатнього освітлення.

Переваги: гнучкість у виборі режиму, цікаве візуальне оформлення, наявність базової освітньої інформації про серцево-судинну систему.

Недоліки: безконтактний режим дуже чутливий до умов освітлення і руху; на пристроях середнього класу можуть спостерігатися відчутні затримки в обробці; функція тривалого моніторингу відсутня; класифікація результатів за категоріями реалізована лише на дуже верхньому рівні без урахування індивідуальних особливостей користувача.

1.4.5 Heart Rate Plus

Heart Rate Plus [12] — безкоштовний застосунок із простим інтерфейсом, який задовольняє базову потребу: виміряти пульс у будь-який момент часу. Цільова аудиторія — користувачі, яким не потрібні розширені функції аналізу.

Переваги: повна безкоштовність, мінімалістичний інтерфейс, швидкий результат.

Недоліки: відсутність детальної історії вимірювань або вона обмежена лише останніми кількома значеннями; немає візуалізації самого PPG-сигналу під час вимірювання; жодних рекомендацій щодо коректного розміщення

пальця або інтерпретації результату; алгоритм підрахунку пульсу спрощений і у складних умовах (слабке освітлення, рух) видає некоректні значення.

1.5 Порівняльний аналіз застосунків-аналогів

Зведене порівняння розглянутих застосунків за основними критеріями функціональності та якості реалізації подано у таблиці 1.2.

Таблиця 1.2 — Порівняння мобільних застосунків

Застосунок	**Платно**	**Історія**	**Графік**	**Категорії**	**Поради**	**Автостарт**
Instant Heart Rate	частково	так	ні	так	базові	ні
Runtastic HR Monitor	частково	так	частково	так	базові	ні
Welltory	частково	так	так	розш.	розш.	ні
Cardiio	частково	так	ні	базові	базові	ні
Heart Rate Plus	ні	обмеж.	ні	ні	немає	ні

Проведений аналіз дозволяє виокремити декілька закономірностей поточного стану ринку. По-перше, переважна більшість продуктів не показує користувачеві форму PPG-сигналу під час вимірювання, обмежуючись анімаційними елементами; це позбавляє користувача наочного підтвердження того, що сигнал коректно зчитується. По-друге, автоматичний старт вимірювання за рівнем яскравості кадру — тобто без необхідності окремо натискати кнопку — реалізований лише в поодиноких застосунках, хоча є очевидним зручним рішенням. По-третє, жоден із розглянутих застосунків не надає інформації про використовуваний алгоритм обробки сигналу, що ускла-

днює об’єктивну оцінку точності для технічно підготовленого користувача. Нарешті, у більшості випадків продукти комерціалізовані через підписку, і користувач, який бажає лише періодично контролювати ЧСС, змушений миритися або з обмеженою функціональністю безкоштовної версії, або з нав’язливою рекламою.

1.6 Постановка задачі та формулювання вимог

З огляду на проведений аналіз, у межах дипломного проєкту розробляється мобільний застосунок, який враховує виявлені недоліки існуючих рішень і пропонує покращений підхід до обробки PPG-сигналу. Основні відмінні риси розроблюваного застосунку:

- **Удосконалений алгоритм обчислення ЧСС** — з використанням червоного каналу у централізованому ROI замість усередненої яскравості кадру, смугової фільтрації у фізіологічному діапазоні частот і обчислення BPM за міждарними інтервалами.
- **Візуалізація PPG-сигналу у реальному часі** — графік форми хвилі під час вимірювання, що дозволяє користувачеві візуально оцінити якість сигналу і скоригувати розміщення пальця.
- **Автоматичний старт вимірювання** — система сама визначає момент прикладення пальця до камери за рівнем яскравості кадру і вмикає вимірювання без потреби у додаткових діях користувача.
- **Категоризація результату** — класифікація отриманого значення ЧСС на категорії «низька», «нормальна» та «висока» згідно з рекомендованими діапазонами для стану спокою.
- **Локальна історія вимірювань** — збереження всіх результатів на пристрої без необхідності у віддаленому сервері та інтернет-з’єднанні.
- **Відсутність прихованої комерціалізації** — застосунок передбачає

весь функціонал доступним без підписки та реклами.

Сукупність цих рішень визначає функціональні вимоги до системи, які детально описані в технічному завданні на дипломний проєкт.

ВИСНОВКИ ДО РОЗДІЛУ 1

У межах першого розділу проведено системний аналіз предметної області дипломного проєкту. Розглянуто медичні та технічні особливості задачі моніторингу частоти серцевих скорочень, обґрунтовано важливість регулярного контролю цього показника як індикатора загального стану здоров'я людини.

Виконано порівняльний огляд основних методів вимірювання ЧСС: електрокардіографії, пульсової оксиметрії, контактної та безконтактної фотоплетизмографії. Показано, що для реалізації побутового мобільного застосунку оптимальним є метод контактної фотоплетизмографії з використанням вбудованої камери смартфона як оптичного датчика, оскільки він поєднує прийнятну точність вимірювання з відсутністю вимог до додаткового обладнання.

Розглянуто фізичні основи та технічні особливості реалізації методу на платформі смартфона: модель Beer-Lambert, специфіку зчитування сигналу з CMOS-сенсорів, вибір колірного каналу та області інтересу, типову послідовність обробки PPG-сигналу та основні джерела похибок вимірювання.

Виконано аналіз п'яти найпопулярніших мобільних застосунків відповідного призначення, наявних на ринку: Instant Heart Rate, Heart Rate Monitor (Runtastic), Welltory, Cardiio та Heart Rate Plus. Виявлено типові недоліки існуючих рішень, серед яких: відсутність візуалізації сигналу у реальному часі, відсутність автоматичного старту вимірювання, переважна модель монетизації через платну підписку.

На основі результатів огляду сформульовано задачу розробки мобільного застосунку, який усуває виявлені недоліки і пропонує покращений алгоритм обчислення ЧСС за рахунок виділення червоного каналу в централізованому ROI, смугової фільтрації сигналу та обчислення BPM за міждар-

ними інтервалами. Деталі архітектури та технологічного стеку розглянуто у наступному розділі.

РОЗДІЛ 2
ОГЛЯД ТЕХНОЛОГІЙ РОЗРОБКИ ЗАСТОСУНКУ

У межах даного розділу обґрунтовується вибір технологічного стеку для реалізації мобільного застосунку моніторингу серцевого ритму. Розглядаються архітектурний підхід, мова програмування, бібліотеки для роботи з камерою та асинхронного програмування, засоби локального зберігання даних, бібліотеки для побудови користувацького інтерфейсу, а також підходи до реалізації алгоритмів цифрової обробки PPG-сигналу.

2.1 Загальна архітектура застосунку

Перед обговоренням окремих технологій варто визначити загальну архітектурну концепцію, у межах якої вони будуть використовуватися. Сучасна розробка мобільних застосунків зазвичай ґрунтується на принципах розділення відповідальності між частинами коду, що відповідають за різні рівні функціональності. Один із найпоширеніших підходів — багатошарова архітектура у вигляді, відомому як *Clean Architecture*, описана Робертом Мартіном [13].

Сутність підходу полягає у виділенні кількох логічних шарів, кожен із яких має чітко визначену зону відповідальності і залежить лише від шарів, розташованих внутрішнє за рівнем абстракції. Для мобільного застосунку прийнято виділяти три основні шари:

- **Presentation layer** — шар представлення, що відповідає за взаємодію з користувачем. Сюди входять Activity, Fragment або відповідні Compose-функції, а також класи, що готують дані для відображення на екрані.
- **Domain layer** — шар бізнес-логіки. Тут розміщуються основні мо-

делі предметної області, інтерфейси репозиторіїв та так звані *use case*-класи — сценарії використання, кожен з яких описує одну дискретну операцію у термінах прикладної задачі (наприклад, «зберегти результат вимірювання», «отримати історію»). Цей шар не залежить ані від платформи, ані від конкретних бібліотек, що робить його придатним для повторного використання та незалежного тестування.

– **Data layer** — шар роботи з даними. Реалізує конкретні шляхи отримання та збереження інформації: робота з базою даних, файловою системою, мережею, апаратними сенсорами. Класи цього шару реалізують інтерфейси репозиторіїв, оголошені у domain layer.

Подібний підхід застосовується у вихідному коді розроблюваного застосунку: пакети `presentation`, `domain` і `data` формують основу проєктної структури. Перевагою такої організації є висока тестопридатність, можливість заміни реалізацій окремих шарів без зміни решти коду, а також зрозуміла структура для подальшої підтримки. Загальна архітектурна схема показана на рис. 2.1.

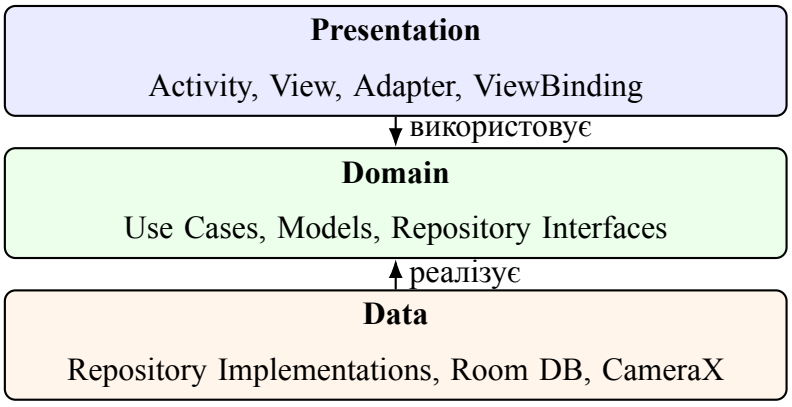


Рисунок 2.1 — Шари архітектури мобільного застосунку

Стрілки на схемі позначають напрям залежностей. Шар Presentation залежить від Domain, але не безпосередньо від Data; натомість Data реалі-

зує інтерфейси, оголошені у Domain. Така схема залежностей гарантує, що бізнес-логіка не «прив'язана» до конкретного фреймворку або платформи.

2.2 Платформа та мова програмування

2.2.1 Платформа Android

Цільовою платформою для розроблюваного застосунку обрано Android, яка станом на 2026 рік займає близько 70 % глобального ринку мобільних операційних систем [14]. Вибір Android зумовлений декількома чинниками: широким парком пристроїв, що охоплює як преміальні флагмани, так і бюджетні моделі, відкритістю екосистеми розробки, наявністю якісного офіційного тулчейну (Android Studio [15], Gradle [16], Android SDK [17]) та зрілої документації [18], а також зручним доступом до апаратних можливостей пристрою, включно з камерами та LED-ліхтариком.

У межах роботи в якості мінімального рівня API обрано 26 (Android 8.0 Oreo), що забезпечує сумісність із пристроями від 2017 року випуску і охоплює близько 95 % активних користувачів Android-екосистеми [19]. Цільовий рівень API — 33 або вище, що відповідає сучасним рекомендаціям Google Play щодо нових публікацій.

2.2.2 Мова програмування Kotlin

Мовою програмування реалізації обрано Kotlin — сучасну статично типізовану мову, розроблену компанією JetBrains і офіційно рекомендовану Google для розробки під Android з 2017 року. Стосовно класичної Java, яка тривалий час була основною мовою платформи, Kotlin має низку суттєвих переваг [20]:

– лаконічніший і більш виразний синтаксис, що знижує обсяг шаблонного коду (boilerplate);

– null-safety на рівні системи типів — мова розрізняє nullable і non-nullable посилання, що дозволяє виявляти більшість NullPointerException на етапі компіляції;
– повноцінна підтримка функціональних конструкцій (лямбди, функції вищого порядку, immutable-колекції);
– вбудовані механізми асинхронного програмування у вигляді coroutines, що уникає типового для Java callback hell;
– стовідсоткова інтероперабельність із Java, що дозволяє використовувати весь накопичений за десятиліття пул бібліотек.

Зокрема для алгоритмів обробки сигналу, які займають центральне місце у даній роботі, властивості Kotlin виявляються особливо зручними: робота з масивами та послідовностями виглядає природно, численні функції стандартної бібліотеки (`map`, `windowed`, `zipWithNext`, `average`) дозволяють описати багато кроків обробки декларативно, без явних циклів та проміжних змінних. Як ілюстративний приклад наведено фрагмент коду, що обчислює ковзне середнє по послідовності значень:

```
val smoothed = rawSignal.windowed(size = 5, step = 1)
                        .map { window -> window.average() }
```

Такий код є водночас лаконічним і виразним: одного погляду достатньо, щоб зрозуміти, що відбувається.

2.3 Робота з камерою: бібліотека CameraX

Доступ до камери на платформі Android історично здійснювався через один з двох API: застарілий Camera1 (deprecated з рівня API 21) та сучасніший Camera2. Camera2 надає максимальний контроль над характеристиками зйомки, але вимагає написання значного обсягу коду для обробки сесій захоплення, обертання, життєвого циклу та поведінки на різних виробниках

пристроїв.

З 2019 року Google пропонує альтернативу — бібліотеку *CameraX* [21], яка є частиною Android Jetpack і надає високорівневий API над Camera2. CameraX забезпечує:

– прозору інтеграцію з життєвим циклом Activity або Fragment через інтерфейс `LifecycleOwner` — автоматичне відкриття та закриття камери у відповідь на події життєвого циклу;
– три основні *use case*-сценарії використання камери: `Preview` для відображення прев'ю на екрані, `ImageAnalysis` для покадрового аналізу і `ImageCapture` для зйомки одиничних знімків. Розробник може комбінувати потрібні сценарії в одній сесії;
– уніфікований інтерфейс роботи з кадрами у вигляді `ImageProxy`, що приховує від розробника відмінності форматів зображення (`YUV_420_888`, `NV21`, `RGBA_8888`) на різних пристроях;
– автоматичну обробку особливостей конкретних виробників та моделей пристроїв (*quirks*), що позбавляє розробника написання обхідних шляхів для нестандартної поведінки камер.

Для задачі реєстрації PPG-сигналу основним сценарієм є `ImageAnalysis`: бібліотека постачає у callback кадри послідовно, з налаштованою стратегією backpressure (наприклад, `STRATEGY_KEEP_ONLY_LATEST`, що відкидає старі кадри, якщо обробка не встигає за камерою). Доступ до пікселів здійснюється через `ImageProxy.planes` — масив байтових буферів для кожного колірного плану. Це дозволяє безпосередньо зчитувати значення інтенсивності з потрібного каналу без проміжного перетворення кадру у Bitmap, що було б непродуктивним при частоті 30 кадрів на секунду.

Окремим аспектом, важливим для PPG, є можливість керування LED-ліхтариком пристрою. CameraX надає для цього метод `CameraControl.enableTorch(true)`, який вмикає ліхтарик на час сесії

камери і автоматично вимикає його при її завершенні. Це робить інтеграцію постійного підсвічування пальця прямолінійною.

2.4 Асинхронне програмування: Kotlin Coroutines і Flow

Обробка відеопотоку з частотою 30 кадрів за секунду є асинхронною задачею, яка не повинна блокувати головний потік інтерфейсу користувача. Тривалий час стандартним підходом до асинхронності на Android були `AsyncTask`, `Handler`, `Looper` та різноманітні callback-механізми, які при ускладненні логіки швидко перетворювали код на «нечитабельну драбину».

Kotlin Coroutines [22] являють собою сучасне рішення цієї задачі: легковагові обчислювальні одиниці, що дозволяють писати асинхронний код у послідовному, синхронному стилі. Корутина може призупинятися (*suspend*) у певних точках виконання, очікуючи на завершення асинхронної операції, і продовжуватися без блокування потоку виконання. Це радикально спрощує читабельність коду і робить очевидним порядок виконання операцій.

Ключові поняття API:

- `CoroutineScope` — область видимості, у межах якої виконуються корутини. Зазвичай прив'язується до життєвого циклу компонента (`lifecycleScope` для Activity), що гарантує автоматичне скасування корутин при знищенні відповідного компонента.
- `Dispatcher` — диспетчер потоків, на яких виконується корутина: `Main` для роботи з UI, `Default` для обчислень, `IO` для блокуючих операцій вводу-виводу.
- `Flow` [23] — декларативне представлення асинхронного потоку значень. На відміну від `Sequence`, який обчислюється у виклику, `Flow` може випускати значення асинхронно з часом, що ідеально підходить для представлення потоку кадрів з камери або потоку результатів вимірювань з бази даних.

У розроблюваному застосунку Flow використовується для двох основних задач: отримання потоку `ImageProxy` від CameraX (через `callbackFlow`) та реактивна доставка історії вимірювань з бази даних на екран історії. Корутини у поєднанні з відповідними диспетчерами забезпечують виконання обробки сигналу на фоновому потоці, тоді як оновлення UI відбувається в Main Dispatcher.

2.5 Локальне сховище даних: Room

Для збереження історії вимірювань ЧСС на пристрої користувача використовується бібліотека Room [24] — офіційний надбудовний шар Android Jetpack над SQLite. Безпосереднє використання SQLite через нативні API Android (`SQLiteDatabase`, `Cursor`, ручне написання запитів) є робочим варіантом, але має суттєві недоліки: значний обсяг шаблонного коду, відсутність перевірки коректності SQL на етапі компіляції, складність підтримки міграцій схеми бази.

Room вирішує ці проблеми за рахунок генерації коду на основі анотацій. Для роботи з базою достатньо описати три основні елементи:

- **Entity** — клас-сутність, що відповідає таблиці у базі даних. Опис здійснюється через анотацію `@Entity(tableName = "...")` над класом, окремі поля позначаються `@PrimaryKey`, `@ColumnInfo` тощо.
- **DAO** — інтерфейс доступу до даних (*Data Access Object*), методи якого позначені анотаціями `@Insert`, `@Update`, `@Delete`, `@Query`. Тіло запитів пишеться як звичайний SQL, але перевіряється компілятором Room на синтаксичну і семантичну коректність відносно описаних Entity.
- **Database** — абстрактний клас, що поєднує Entity і DAO в одну базу. Реалізація автоматично генерується Room на етапі компіляції.

У застосунку для збереження результатів вимірювань використовується одна таблиця `results` зі стовпцями `id` (первинний ключ з автогенерацією), `result` (числове значення ЧСС), `time` і `date` — текстові поля для зручного відображення. Room підтримує повернення даних у вигляді `Flow<List<Result>>`, що автоматично оновлює підписників при зміні вмісту таблиці — це використовується для реактивної доставки історії на екран.

2.6 Користувацький інтерфейс

2.6.1 Розмітка інтерфейсу та View Binding

Для побудови користувацького інтерфейсу обрано класичний підхід на базі XML-розмітки та View-системи Android, з активним використанням бібліотеки *View Binding* [25]. View Binding автоматично генерує клас-зв'язок для кожного XML-файлу розмітки, надаючи типобезпечний доступ до елементів інтерфейсу. Це дозволяє уникнути виклику `findViewById(...)` вручну і типічних помилок неузгодженості ідентифікаторів між кодом і розміткою.

Альтернативою була б декларативна побудова інтерфейсу на базі Jetpack Compose [26] — сучасного UI-фреймворку для Android. Compose має чимало переваг (декларативність, реактивність, інтеграція з Kotlin), однак на момент початку розробки даного проєкту класичний підхід через XML-розмітку залишався більш стабільним і вимагав меншого порогу входження, що було важливо для зосередження зусиль на основній технічній задачі — алгоритмі обробки PPG-сигналу. Тому як основний інструмент обрано саме XML+View Binding.

2.6.2 Анімація: бібліотека Lottie

Для надання інтерфейсу візуальної привабливості, особливо на ключових екранах (заставка, екран вимірювання), використовується бібліотека Lottie [27] від компанії Airbnb. Lottie дозволяє відтворювати анімації у форматі JSON, експортованому з Adobe After Effects через плагін Bodymovin. Перевагою такого підходу порівняно з традиційними способами анімації (PNG-послідовності, анімовані GIF, ручні Canvas-анімації) є малий розмір файлів, векторна природа анімацій (їх можна довільно масштабувати без втрати якості) та можливість програмного керування ходом відтворення (швидкість, цикли, проміжні стани).

У межах розроблюваного застосунку Lottie виконує дві ролі. По-перше, керує тривалістю сеансу вимірювання: спеціально підібрана JSON-анімація відтворюється з фіксованою швидкістю, а її завершення є тригером переходу до екрана результату. Перевагою саме цього механізму є вбудовані можливості `pauseAnimation()` та `resumeAnimation()`, які дозволяють тимчасово зупиняти таймер у разі втрати контакту пальця з лінзою і поновлювати його після повернення — тим самим забезпечуючи, що повний бюджет вимірювання отримує лише чистий PPG-сигнал. По-друге, Lottie може бути використана для відтворення складніших дизайнерських анімацій у майбутніх версіях. Основні візуальні ефекти інтерфейсу (дихаюче серце на головному екрані, концентричні pulse-rings та анімовані стовпчики хвилі на екрані вимірювання, лічильник BPM на екрані результату) реалізовані нативно через `ObjectAnimator` і `ValueAnimator` — це енергоефективніше і не потребує підготовки сторонніх JSON-файлів. Це не лише прикрашає інтерфейс, але й виконує функціональну роль — надає користувачеві зворотний зв’язок про те, що процес триває.

2.7 Засоби цифрової обробки сигналу

Алгоритмічна частина роботи — обчислення ЧСС за PPG-сигналом — вимагає реалізації певних класичних методів цифрової обробки сигналу. На цьому етапі розглянемо доступні підходи та обґрунтуємо вибір конкретних методів.

2.7.1 Видалення тренду базової лінії

Базова лінія PPG-сигналу повільно дрейфує з часом, що зумовлено як фізіологічними чинниками (зміна тиску пальця на камеру, дихальні рухи), так і технічними (тепловий дрейф сенсора камери). Існує декілька підходів до видалення цього тренду:

- **Відняття ковзного середнього** — найпростіший метод. З сигналу віднімається його значення, усереднене за вікном певної довжини. Перевага — висока швидкість обчислення; недолік — внесення фазового зсуву у частотну характеристику.
- **Поліноміальна апроксимація** — сигнал апроксимується поліномом невисокого порядку (зазвичай 3–5), який потім віднімається. Підходить для видалення гладких трендів.
- **Метод Tarvainen** [7] — спеціально розроблений для біомедичних сигналів, поєднує згладжування з штрафним членом за нерівність. Дає високу якість результату, але обчислювально дорожчий.

Для задачі обробки PPG-сигналу у реальному часі на мобільному пристрої обрано перший варіант — відняття ковзного середнього з вікном порядку 1–2 с. Цього достатньо для усунення дрейфу, а обчислювальна вартість мінімальна.

2.7.2 Смугова фільтрація

Фізіологічний діапазон частот ЧСС у стані спокою та помірних навантажень становить приблизно 0,7–3,5 Гц (відповідає 42–210 уд/хв). Для виокремлення лише корисних коливань у цьому діапазоні застосовується смуговий цифровий фільтр. Серед можливих варіантів реалізації:

- **Каскад однополюсних IIR-фільтрів** (*Infinite Impulse Response*) — послідовне з'єднання простого high-pass і low-pass фільтрів першого порядку. Кожен фільтр описується одним коефіцієнтом α і одним елементом пам'яті. Сумарний спад — близько -12 дБ/октаву; амплітудно-частотна характеристика має незначний spad у смузі пропускання, але проста у реалізації та має мінімальну обчислювальну вартість.
- **Фільтр Баттерворта** — класичний IIR-фільтр з *максимально плаского* амплітудно-частотною характеристикою у смузі пропускання. Для еквівалентної крутизни потребує другого порядку (5 коефіцієнтів і 2 елементи пам'яті), має комплексні полюси та забезпечує точніший passband.
- **Фільтр Чебишева** — допускає пульсації у смузі пропускання або в смузі затримки в обмін на крутіший спад. Для нашої задачі рівне підсилення у смузі важливіше за крутий спад.
- **FIR-фільтр** — скінченний імпульсний фільтр. Дає лінійну фазу, але потребує більшого порядку для еквівалентної крутизни і відповідно більше обчислень.

Зважаючи на вимоги до якості й обчислювальної ефективності, а також на вузькосмуговий характер самого PPG-сигналу (домінантна гармоніка зосереджена близько 1–1,5 Гц), для реалізації обрано **каскадний IIR-фільтр першого порядку** (high-pass з частотою зрізу 0,7 Гц у поєднанні з low-pass з

частотою зрізу 3,5 Гц) [28; 29]. Цей варіант поступається повноцінному Баттерворту другого порядку у пласкості АЧХ у смузі пропускання, проте для вузькосмугового PPG-сигналу різниця у точності оцінки ВРМ є несуттєвою, а реалізація — лаконічною (по одному коефіцієнту на каскад) і обчислювально дешевою. Коефіцієнти α_{HP} і α_{LP} розраховуються одноразово при ініціалізації обробника для заданої частоти дискретизації (приблизно 30 Гц — частота кадрів камери) за класичними формулами білінійної апроксимації RC-кола.

2.7.3 Детекція піків

Виявлення локальних максимумів у відфільтрованому PPG-сигналі є ключовою операцією для подальшого розрахунку міжударних інтервалів. Базовий алгоритм детекції перевіряє три послідовні точки сигналу: якщо центральна з них перевищує обидві сусідні, вона є локальним максимумом. На практиці цей наївний підхід поєднується з додатковою умовою — **обмеженням мінімального інтервалу між піками**: після виявлення піку наступний пік шукається не раніше ніж через певний час (наприклад, 0,35 с, що відповідає максимально можливій ЧСС близько 170 уд/хв). Це відсікає подвоєння одного серцевого скорочення в окремих випадках форми хвилі. У разі коли два піки знайдено надто близько один до одного, зберігається лише вищий за амплітудою.

Реалізація з цим доповненням у Kotlin виглядає лаконічно і виконується за один прохід через масив сигналу.

2.7.4 Альтернативний підхід: FFT

Окрім аналізу у часовій області шляхом детекції піків, існує альтернативний шлях — частотний аналіз сигналу через швидке перетворення Фур'є (FFT, *Fast Fourier Transform*). Ідея полягає у перетворенні сигналу в спектр потужності і пошуку домінантної частоти у смузі ЧСС, яка і відповідатиме

шуканому значенню BPM.

Перевагою FFT є відносна стійкість до окремих артефактів та відсутність потреби у точному знаходженні піків. Однак FFT вимагає достатньо тривалого запису (не менше 8–10 с) для прийнятної частотної роздільної здатності і має нижчу часову локалізацію — одне значення ЧСС обчислюється на весь сегмент. Аналіз у часовій області, навпаки, дає по одному значенню на кожне серцеве скорочення, що дозволяє оцінити варіабельність ритму (HRV) і реагувати на короткочасні зміни.

Для розроблюваного застосунку обрано аналіз у часовій області з детекцією піків та обчисленням BPM через IBI, як описано у попередньому розділі. Підхід FFT може бути доданий як альтернативний алгоритм у майбутніх версіях.

ВИСНОВКИ ДО РОЗДІЛУ 2

У межах другого розділу розглянуто та обґрунтовано вибір технологій, на яких будується розроблюваний мобільний застосунок.

В якості загальної архітектурної концепції обрано підхід багатошарової архітектури (Clean Architecture) з виділеними шарами presentation, domain і data. Такий підхід забезпечує розділення відповідальності, високу тестопридатність та зрозумілу організацію кодової бази.

Цільовою платформою для реалізації застосунку обрано Android з мінімальним рівнем API 26 (Android 8.0). Як мову програмування використано Kotlin, що забезпечує лаконічність коду, типобезпечність та зручну інтеграцію із сучасним стеком Android-розробки.

Для роботи з відеопотоком камери та керування LED-ліхтариком обрано бібліотеку CameraX, яка забезпечує високорівневий API та автоматичну інтеграцію з життєвим циклом компонента. Для асинхронної обробки кадрів використовуються Kotlin Coroutines і Flow, що дозволяють писати асинхронний код у послідовному стилі без використання callback-механізмів.

Збереження історії вимірювань на пристрої реалізовано через бібліотеку Room — надбудову над SQLite з типобезпечним API і компіляційною перевіркою SQL-запитів. Користувацький інтерфейс будується на базі класичної XML-розмітки з View Binding; анімаційні елементи інтерфейсу реалізовані нативно через `ObjectAnimator` і `ValueAnimator`, а бібліотека Lottie підключена як резервний інструмент відтворення JSON-анімацій.

Розглянуто алгоритмічні підходи до цифрової обробки PPG-сигналу: методи видалення тренду базової лінії, варіанти смугової фільтрації, алгоритми детекції піків та частотний аналіз через FFT. Для реалізації обрано: відняття ковзного середнього для видалення тренду, каскадний IIR-фільтр першого порядку (high-pass з частотою зрізу 0,7 Гц та low-pass з частотою зрізу

3,5 Гц) для смугової фільтрації у діапазоні 0,7–3,5 Гц, а також детекцію піків з порогом амплітуди та мінімальним міжпіковим інтервалом з подальшим обчисленням ВРМ через міждарні інтервали.

Сукупність обраних технологій формує самодостатню технологічну базу, у межах якої побудовано архітектуру застосунку та реалізовано основну функціональність. Деталі програмної реалізації, структура проєкту та модулі описані у наступному розділі.

РОЗДІЛ 3
ДЕТАЛІ РОЗРОБКИ ЗАСТОСУНКУ

У цьому розділі розглядаються конкретні рішення, прийняті під час реалізації мобільного застосунку: структура файлів проєкту, схема зберігання даних, реалізація модулів захоплення та аналізу відеопотоку, повний цикл обчислення BPM з PPG-сигналу, а також особливості користувацького інтерфейсу.

3.1 Структура проєкту

Кодова база застосунку [30] організована відповідно до принципів Clean Architecture, описаних у підрозділі 2.1. У межах одного Android-модуля код розподілено за трьома пакетами, що відповідають архітектурним шарам.

Пакет `presentation` містить компоненти користувацького інтерфейсу: класи Activity (`MainActivity`, `OnboardingActivity`, `HomeActivity`, `InstructionActivity`, `MeasureActivity`, `ResultActivity`, `HistoryActivity`), Fragment (`OnboardingFragment`), а також допоміжне розширення `WindowInsetsExt` для адаптивних відступів. До цього шару підключаються бібліотеки View Binding та ViewPager2 [31].

Пакет `domain` містить бізнес-логіку і не залежить від Android-фреймворку. Сюди входять: модель даних `Result`, інтерфейси репозиторіїв (`CameraRepository`, `ResultRepository`, `TimeRepository`), use case-класи (додавання результату, отримання історії, очищення історії, запуск камери), а також ключові обчислювальні модулі — `PpgProcessor` і `FingerDetector`. Незалежність цього шару від Android робить можливим його юніт-тестування у подальших ітераціях розвитку проєкту.

Пакет `data` містить реалізації репозиторіїв (`CameraRepositoryImpl`

на основі CameraX, `ResultRepositoryImpl` на основі Room, `TimeRepositoryImpl` на основі стандартних класів Java для роботи з часом), визначення бази даних (`MainDb`, інтерфейс `Dao`) та адаптер `ResultAdapter` для відображення історії у RecyclerView [32].

Ресурси інтерфейсу (XML-розмітки екранів, drawable-ресурси, кольори, рядки, launcher icon) розміщені у стандартній структурі теки `res` проєкту Android.

Таке розбиття дозволяє замінити окремі реалізації без впливу на бізнес-логіку: наприклад, у тестовому оточенні `CameraRepository` можна замінити фейковою реалізацією, яка віддає попередньо записаний відеопотік, без жодних змін у `MeasureActivity` чи `PpgProcessor`.

3.2 Модель даних і локальне сховище

Кожне вимірювання представлено моделлю `Result`, яка містить обчислене значення BPM, час і дату вимірювання, а також унікальний ідентифікатор (генерується автоматично при вставці у базу).

Доступ до бази даних реалізовано через бібліотеку Room — надбудову над SQLite з типобезпечним API та компіляційною перевіркою SQL-запитів. Інтерфейс `Dao` оголошує три операції: вставку нового запису (*suspend*-функція `insertResult`), отримання всіх записів у вигляді `Flow<List<Result>>` з сортуванням за спаданням ідентифікатора, а також повне очищення таблиці (`deleteAllResults`). Тип `Flow` є ключовим: він автоматично сповіщає підписників про будь-які зміни у базі, тому інтерфейс історії оновлюється без явних викликів — наприклад, після завершення вимірювання нова картка з’являється у списку миттєво.

База даних інкапсулюється у singleton-класі `MainDb`, який використовує патерн *double-checked locking* для забезпечення єдиного екземпляра впродовж усього життєвого циклу процесу застосунку. На рівні домену взає-

модія з базою відбувається виключно через інтерфейс `ResultRepository` і відповідні use case-класи (`AddResultUseCase`, `ShowHistoryUseCase`, `ClearHistoryUseCase`), які не знають про деталі реалізації сховища і працюють з абстракцією домену.

3.3 Реалізація захоплення відеопотоку

3.3.1 Налаштування CameraX

Захоплення відеопотоку реалізовано через бібліотеку CameraX, обрану і обґрунтовану у підрозділі 2.3. Камера прив'язується до життєвого циклу Activity через `ProcessCameraProvider`, що автоматично звільняє ресурси при паузі чи знищенні. Для отримання кадрів використовується `ImageAnalysis` зі стратегією `STRATEGY_KEEP_ONLY_LATEST` — вона пропускає накопичені кадри, якщо обробка не встигає за швидкістю потоку, що гарантує відсутність зростаючої затримки у буфері.

Власне потік кадрів представлений у вигляді Kotlin Flow: відповідний метод репозиторію використовує `callbackFlow` для перетворення асинхронного callback-механізму CameraX у послідовний потік. Кадри постачаються через окремий `ExecutorService`, виділений під аналіз. Посилання на отриманий об'єкт `Camera` зберігається у поле класу для подальшого керування ліхтариком через `cameraControl.enableTorch(true)`.

3.3.2 Виділення області інтересу (ROI)

Кожен отриманий кадр у форматі YUV_420_888 не обробляється повністю — усереднення значень кольорових каналів проводиться лише по центральних $50\,\% \times 50\,\%$ пікселів. Це дозволяє відсікти крайові артефакти (віньєтування, фокусні нерівності) та зменшити кількість операцій у чотири рази у порівнянні з повним кадром. На типовому роздільному 480p ROI містить

близько 6 000–8 000 точок, чого більш ніж достатньо для подавлення теплового шуму сенсора шляхом усереднення. Для додаткового прискорення обхід ведеться з кроком 2 пікселі по обидвох осях.

3.3.3 Перетворення кольорового простору YUV у RGB

Сирим PPG-сигналом для подальшого аналізу є середнє значення червоного (R) каналу по ROI. Оскільки CameraX надає кадри у форматі YUV (окремі площини з компонентами Y, U, V), необхідно перевести ці компоненти до простору RGB за формулами стандарту ITU-R BT.601:

$$\begin{aligned} R &= Y + 1{,}402 \cdot (V - 128), \\ G &= Y - 0{,}344 \cdot (U - 128) - 0{,}714 \cdot (V - 128), \\ B &= Y + 1{,}772 \cdot (U - 128). \end{aligned} \tag{3}$$

Для коректного індексування пам'яті враховуються особливості розкладу YUV-планів: V- і U-плани мають у два рази меншу просторову роздільну здатність (4:2:0 subsampling), тому індекс пікселя у них обчислюється з ділення координат на два, помноженого на крок рядка (`rowStride`) і крок пікселя (`pixelStride`). Реалізовано два методи: один повертає лише усереднене R-канал (швидший, використовується у вимірюванні), другий — усі три канали R, G, B (для модуля виявлення пальця).

3.4 Реалізація виявлення пальця на лінзі

Перш ніж починати аналіз PPG, необхідно переконатись, що користувач справді приклав палець до об'єктива камери, а не тримає її у повітрі. Простої перевірки яскравості Y-каналу недостатньо: з ввімкненим ліхтариком зображення пальця має не темну, а червонувату середню яскравість (Y близько 50–90), оскільки світло частково проходить крізь тканину пальця.

Натомість надійний критерій будується на специфічній колірній сигнатурі: при освітленні пальця ліхтариком гемоглобін крові поглинає зелено-блакитну частину спектру значно сильніше за червону, тому до сенсора повертається переважно червоне світло. У такому випадку червоний канал значно домінує над зеленим і синім.

Для надійної роботи у різних умовах освітлення та автоекспозиції у класі `FingerDetector` реалізовано два рівні чутливості з *гістерезисом*. Суворий тест використовується для першого підтвердження пальця на екрані інструкції (необхідно гарантувати, що це справді палець, а не випадковий червоний об'єкт у кадрі). М'який тест використовується для безперервного моніторингу під час вже розпочатого вимірювання (палець уже підтверджено, потрібно лише відстежувати, чи не зник він). Конкретні значення порогів подано у таблиці 3.1.

Таблиця 3.1 — Пороги виявлення пальця у FingerDetector

Тест	$R_{\min}$	$R/G_{\min}$	$R/B_{\min}$
Суворий (*strict*)	110	1,4	1,4
М'який (*lenient*)	90	1,15	1,15

Менш суворі пороги м'якого тесту запобігають миготінню стану "палець знятий" через короткочасні коливання автоекспозиції, коли користувач реально продовжує тримати палець.

На екрані інструкції додатково використовується вікно з декількох послідовних кадрів: перехід до екрана вимірювання відбувається лише після того, як палець був підтверджений у ≥ 15 кадрах підряд (приблизно 0,5 с при 30 fps), і лише після пропуску перших 30 кадрів *warmup*-у (поки автоекспозиція камери стабілізується).

3.5 Реалізація алгоритму обчислення ЧСС

Серцевим компонентом застосунку є клас `PpgProcessor`, у якому реалізована описана у підрозділі 2.7 повна послідовність обробки сирого PPG-сигналу. Клас не має жодних залежностей від Android-фреймворку (лише стандартна бібліотека Kotlin), що дозволяє у майбутньому покривати його юніт-тестами без потреби в емуляторі чи реальному пристрої.

3.5.1 Накопичення семплів

Модуль зберігає список пар (*часова мітка*, *значення R-каналу*) у структурі даних `List<Sample>`. Семпли додаються лише тоді, коли `FingerDetector` підтверджує палець у поточному кадрі. Якщо палець на короткий момент зник, семпли пропускаються — це запобігає засміченню сигналу шумом без пальцевої сигнатури.

3.5.2 Оцінка частоти дискретизації

Перед обробкою сигналу необхідно визначити фактичну частоту дискретизації f_s. Наївне обчислення $f_s = N/(t_{last} - t_{first})$ занижує значення, якщо в потоці семплів були паузи (вони збільшують знаменник, але не чисельник). Це у свою чергу зсуває параметри bandpass-фільтра і мінімального міжпікового інтервалу, призводячи до завищення BPM.

Для коректної оцінки використовується медіана міжсемплових інтервалів, з фільтрацією значень > 200 мс (які відповідають паузам, коли палець був тимчасово знятий). Така оцінка стійка до довільної кількості пауз будь-якої тривалості.

3.5.3 Видалення тренду базової лінії і смугова фільтрація

Видалення тренду реалізується відніманням центрованого ковзного середнього з вікном, що дорівнює одній секунді сигналу. Після цього застосовується каскадний IIR-фільтр першого порядку: спочатку high-pass з частотою зрізу 0,7 Гц, далі low-pass з 3,5 Гц. Коефіцієнти α_{HP} і α_{LP} обчислюються один раз для конкретного значення f_s за класичними формулами білінійної апроксимації RC-кола:

$$\alpha_{\text{HP}} = \frac{RC_{\text{HP}}}{RC_{\text{HP}} + \Delta t}, \quad \alpha_{\text{LP}} = \frac{\Delta t}{RC_{\text{LP}} + \Delta t}, \tag{4}$$

де $RC_{\text{HP}} = 1/(2\pi f_{\text{low}})$, $RC_{\text{LP}} = 1/(2\pi f_{\text{high}})$, $\Delta t = 1/f_s$, $f_{\text{low}} = 0{,}7$ Гц, $f_{\text{high}} = 3{,}5$ Гц.

3.5.4 Контроль якості сигналу

Перед детекцією піків здійснюється перевірка амплітуди відфільтрованого сигналу через стандартне відхилення сигналу після видалення тренду. Якщо $\sigma < 0{,}3$, це означає що сигнал занадто плоский — ймовірно, палець був прикладений зі слабким тиском (без видимої пульсації крові) або контакт був нестабільним. У такому випадку метод повертає 0, який інтерпретується UI як “не вдалось виміряти” і відображається символом “—” замість числа ВРМ. Це попереджує демонстрацію користувачеві випадкового числа, отриманого з шумного сигналу.

3.5.5 Детекція піків і обчислення ВРМ

Локальні максимуми у відфільтрованому сигналі шукаються наївним тестом трьох послідовних точок ($x_{i-1} < x_i > x_{i+1}$) з додатковою умовою мінімальної відстані між піками $T_{\text{min}} = 0{,}35$ с (при поточному f_s). Якщо два піки знайдено надто близько — зберігається лише вищий з них.

Із масиву моментів часу піків $\{t_k\}$ обчислюються міжударні інтервали $\mathrm{IBI}_k = t_{k+1} - t_k$, а фінальний BPM визначається за формулою

$$\mathrm{BPM} = \frac{60\,000}{\mathrm{median}(\{\mathrm{IBI}_k\})}, \tag{5}$$

де median — медіана у мілісекундах. Використання медіани, а не середнього, забезпечує стійкість до поодиноких помилкових або пропущених піків. Результуюче значення обмежується фізіологічним діапазоном 40–200 BPM, що захищає від артефактів на межах сигналу.

Узагальнено, обчислення BPM у модулі `PpgProcessor` відбувається у такій послідовності: спочатку перевіряється, що зібрано достатньо семплів (не менше 30) і що тривалість збору перевищує 5 с, інакше повертається 0. Далі обчислюється частота дискретизації f_s як обернена медіана міжсемплових інтервалів (з фільтрацією значень > 200 мс, що відповідають паузам). Після цього виконується видалення тренду відніманням ковзного середнього і перевірка стандартного відхилення сигналу: якщо $\sigma < 0{,}3$, результат теж 0 (сигнал плоский, недостовірний). До отриманого сигналу застосовується каскадний IIR-фільтр (0,7–3,5 Гц) за (4), після чого шукаються локальні максимуми з мінімальною відстанню $0{,}35 \cdot f_s$. Якщо знайдено менше трьох піків — повертається 0. У іншому випадку обчислюються міжударні інтервали і фінальний BPM за формулою (5), обмежений діапазоном 40–200.

3.6 Реалізація користувацького інтерфейсу

Користувацький інтерфейс побудовано на класичних XML-розмітках з використанням ConstraintLayout [33], шрифтового сімейства Rubik і палітри в стилі *calm wellness*: кораловий акцент (#C97163) на кремовому фоні (#F4EDE5), шавлієвий (#6B9C7E) для статусу “у нормі”, бурштиновий (#D9A877) для “уповільнено”/“прискорено” і ліловий (#9B7BAB) для додаткових акцентів. Загалом застосунок містить сім екранів, опис кожного з ілю-

страцією наведено у розділі 4 (підрозділ “Опис екранів застосунку”).

З технічного боку реалізація UI спирається на: ConstraintLayout для всіх екранів, ViewBinding для типобезпечного доступу до View з Kotlin-коду, ViewPager2 з FragmentStateAdapter для онбордингу, RecyclerView з LinearLayoutManager для історії, CardView з `cardCornerRadius` для отримання круглого вигляду PreviewView. Анімації реалізовані переважно нативно через `ObjectAnimator`, `ValueAnimator` і `PropertyValuesHolder` — такий підхід дає плавне, енергоефективне відтворення без потреби в Lottie-файлах для простих сценаріїв (breathing heart на головному екрані, pulse rings та waveform на екрані вимірювання, counter-анімація BPM на результаті, fade-in/out overlay при втраті пальця).

Усі екрани адаптуються до різних висот статус-бара завдяки розширенню View `applyStatusBarTopPadding(extraDp)`, яке встановлює відступ зверху рівним висоті системного статус-бара плюс заданий додатковий запас. Це позбавляє від хардкоду вертикальних відступів і забезпечує однакове візуальне розташування елементів на різних моделях смартфонів.

ВИСНОВКИ ДО РОЗДІЛУ 3

У третьому розділі описано конкретні рішення, прийняті під час реалізації мобільного застосунку Pulsa.

Кодова база розподілена за трьома пакетами відповідно до шарів Clean Architecture: presentation, domain і data. Це забезпечує тестопридатність бізнес-логіки, можливість зміни реалізації окремих компонентів без впливу на інші шари та зрозумілу організацію структури файлів проєкту.

Збереження історії вимірювань реалізовано через бібліотеку Room з типобезпечним SQLite-API. Доступ до бази відбувається лише через інтерфейс репозиторію та use-case класи, що приховує деталі зберігання від presentation-шару.

Захоплення відеопотоку побудовано на бібліотеці CameraX. Реалізовано усереднення значень кольорових каналів по центральній області кадру (ROI 50 % × 50 %) з перетворенням кольорового простору YUV у RGB. Посилання на об’єкт камери зберігається для подальшого керування ліхтариком.

Розроблено окремий модуль `FingerDetector`, який визначає наявність пальця на лінзі за специфічною колірною сигнатурою — домінуванням червоного каналу над зеленим і синім. Два рівні чутливості з гістерезисом запобігають миготінню стану “палець знятий” через коливання автоекспозиції камери.

Реалізовано модуль `PpgProcessor` — клас із повною послідовністю обробки PPG-сигналу: накопичення семплів, стійка до пауз оцінка частоти дискретизації через медіану міжсемплових інтервалів, усунення тренду ковзним середнім, каскадний IIR-фільтр першого порядку, перевірка якості сигналу через стандартне відхилення, виявлення піків з мінімальним інтервалом і обчислення BPM як 60 000/median(IBI). Модуль не залежить від Android-фреймворку.

Користувацький інтерфейс реалізовано на ConstraintLayout з ViewBinding, ViewPager2 для онбордингу і RecyclerView для історії. Усі анімаційні ефекти побудовано нативно через `ObjectAnimator` і `ValueAnimator`, без надмірного використання сторонніх бібліотек. Адаптивне обчислення верхнього відступу гарантує коректне відображення на різних моделях смартфонів. Детальний опис кожного з семи екранів зі скріншотами наведено у розділі 4.

Сукупно реалізована функціональність відповідає вимогам, поставленим у розділі 1, та використовує технології, обґрунтовані у розділі 2.

РОЗДІЛ 4
ДОСЛІДЖЕННЯ ТА АНАЛІЗ РОЗРОБЛЕНОГО ЗАСТОСУНКУ

У четвертому розділі представлено результати дослідження розробленого застосунку: опис кожного з реалізованих екранів, типові сценарії взаємодії користувача, експериментальна перевірка точності обчислення ЧСС у порівнянні з медичним приладом, аналіз продуктивності та обговорення наявних обмежень.

4.1 Опис екранів застосунку

Застосунок Pulsa містить сім екранів, що покривають повний цикл взаємодії з користувачем: від першого знайомства до перегляду історії вимірювань.

Splash — стартовий екран із логотипом застосунку (стилізоване серце у м’якому кораловому крузі), назвою Pulsa і теглайном “Турбота про серце”. Унизу — три точки-індикатори, які по черзі підсвічуються (анімація alpha з затримкою 250 мс між точками). Через приблизно 3 секунди відбувається автоматичний перехід до онбордингу (для першого запуску) або одразу до головного екрана (рис. 4.1).

Рисунок 4.1 — Splash-екран Pulsa

Онбординг — два слайди, реалізовані через ViewPager2. Перший слайд знайомить з ідеєю PPG-вимірювання за допомогою камери смартфона; другий — з можливістю автоматичного збереження історії вимірювань (рис. 4.2). Унизу екрана — кнопка “Далі” (на другому слайді змінюється на “Почати”) і дві точки-індикатори поточної позиції у послідовності. Перехід між слайдами можливий як натисканням кнопки, так і свайпом.

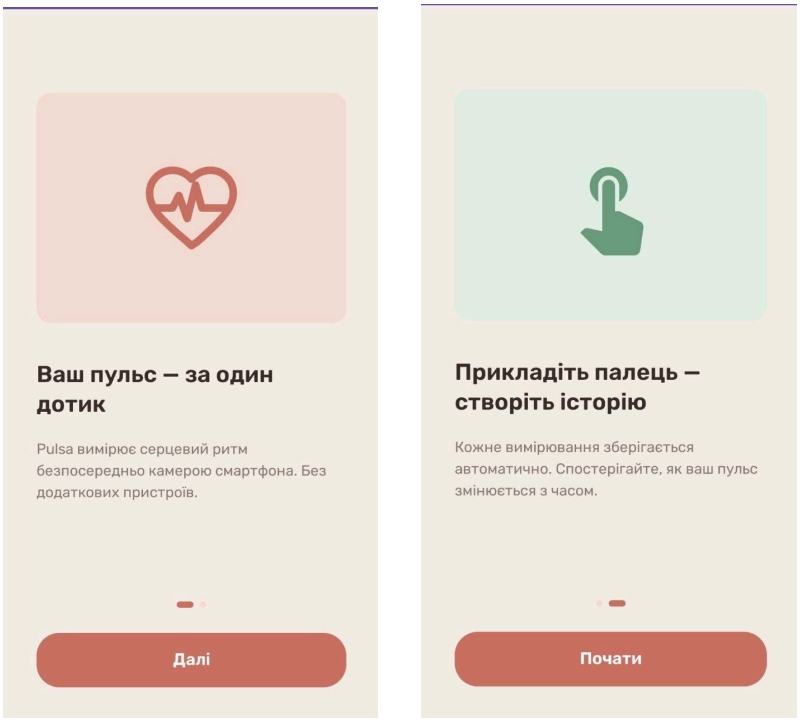


Рисунок 4.2 — Слайди онбордингу: перший і другий

Головна (порожній стан) — основний екран до першого вимірювання. Назва застосунку у лівому верхньому куті, кнопка переходу до історії у правому верхньому куті. По центру — коралове коло з зображенням серця, що плавно “дихає” (масштабується від 1,0 до 1,08 з періодом 1,5 с) у супроводі заголовка “Готові до першого вимірювання?” та інструкції “Прикладіть палець до камери і утримуйте 5–10 секунд”. У нижній частині — велика кругла коралова кнопка з іконкою play для запуску вимірювання (рис. 4.3).

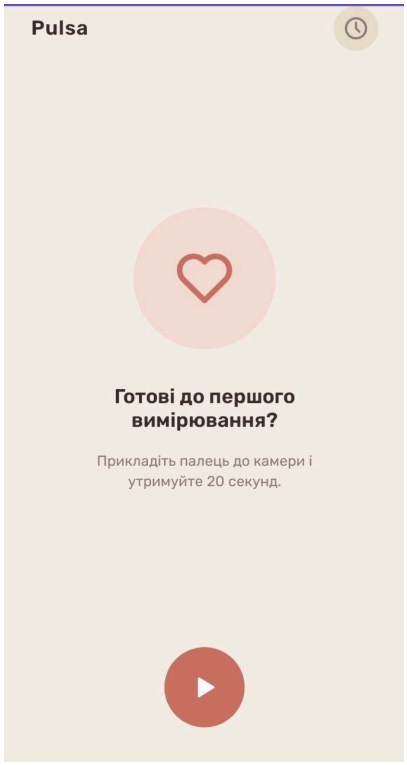


Рисунок 4.3 — Головний екран до першого вимірювання

Інструкція — екран з активованою камерою у стані очікування пальця. У центрі — круглий PreviewView (вікно з поточним зображенням з камери), що дозволяє користувачеві безпосередньо побачити, чи камера працює і як саме виглядає зображення (на рис. 4.4 камера показує оточення — клавіатуру — бо палець ще не прикладено). Нижче — заголовок “Палець на камері” з пояснювальним підписом. У нижній частині — велика ілюстративна картка з зображенням пальця, що допомагає інтуїтивно зрозуміти очікувану позицію. Як тільки система впевнено виявляє палець (R-канал перевищує поріг протягом 15 кадрів поспіль після прогріву автоекспозиції), відбувається автоматичний перехід до екрана вимірювання.

Рисунок 4.4 — Екран інструкції з активною камерою

Активне вимірювання — центральний екран процесу збору PPG-сигналу. По центру — BPM-ядро: кораловий круг з поточною проміжною оцінкою пульсу, навколо якого розходяться концентричні pulse rings (анімація розширення з 0,85 до 1,6 масштабу з одночасним загасанням прозорості від 0,55 до 0). Нижче — текст “Йде вимірювання”/“Утримуйте палець нерухомо”, три точки-індикатори тривалості процесу, що по черзі спалахують, та хвиля з семи стовпчиків, висоти яких анімовано пульсують (візуальна метафора активного аналізу сигналу), див. рис. 4.5. На це накладена система безперервного контролю наявності пальця: якщо палець тимчасово знятий — з’являється overlay з повідомленням “Поверніть палець на камеру”, таймер вимірювання ставиться на паузу, і відновлюється лише після повернення пальця (рис. 4.6). Тобто 20-секундний бюджет вимірювання гарантовано перебувається саме у стані активного контакту з лінзою.

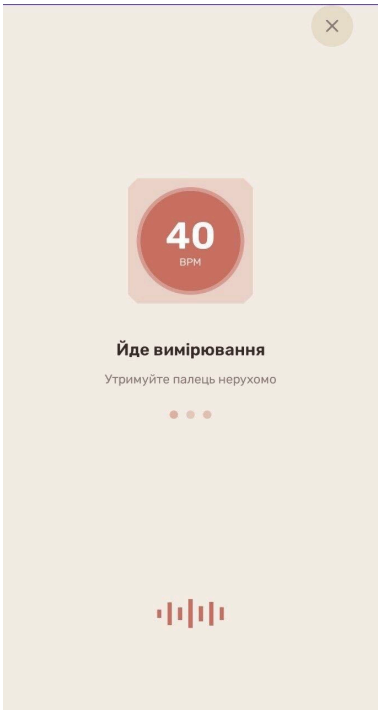


Рисунок 4.5 — Активне вимірювання з відображенням поточного BPM

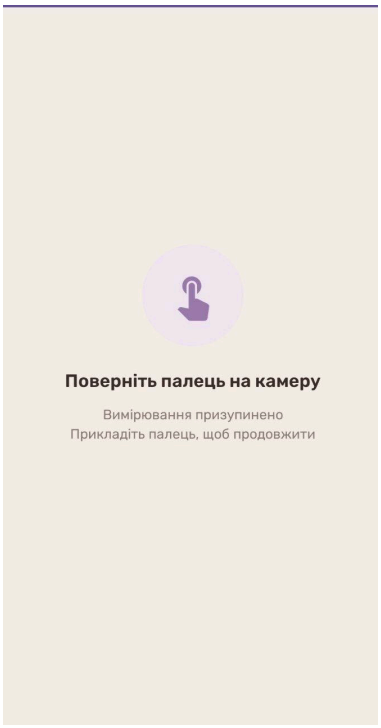


Рисунок 4.6 — Overlay при втраті пальця на лінзі камери

Результат — екран демонстрації обчисленого значення ЧСС. У білій картці посередині — велике число BPM (анімація лічильника від 0 до

фінального значення за 1,4 с з decelerate-інтерполятором), pill-бейдж зі словесним статусом (“Уповільнено”/“У нормі”/“Прискорено” з кольоровим маркуванням бурштин / шавлія / корал відповідно), а також горизонтальна шкала 40–140 BPM з кружком-позначкою, що візуально позначає, де у фізіологічному діапазоні знаходиться отриманий результат (рис. 4.7). Унизу — коралова кнопка “Готово” для повернення на головний екран. Результат автоматично записується у локальну базу даних.

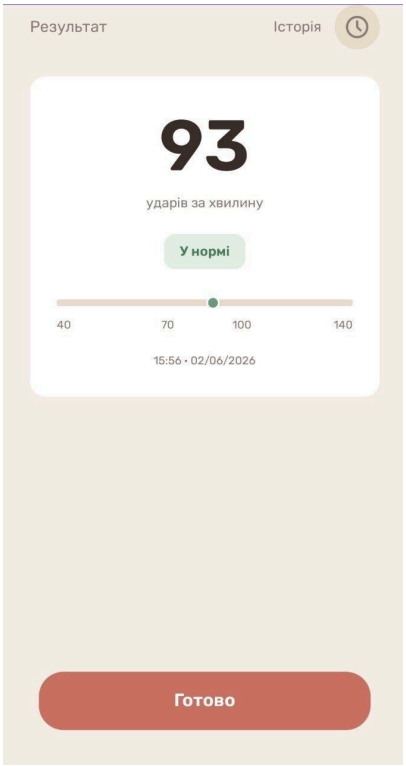


Рисунок 4.7 — Екран результату вимірювання

Історія — список усіх раніше виконаних вимірювань, відсортованих за спаданням часу. Кожна картка містить кольоровий вертикальний маркер ліворуч (бурштиновий / шавлієвий / кораловий залежно від BPM-діапазону), значення BPM великим шрифтом, дату і час вимірювання меншим шрифтом. У нижній частині екрана — коралова кнопка очищення історії, яка з’являється після прокручування до кінця списку (рис. 4.8).

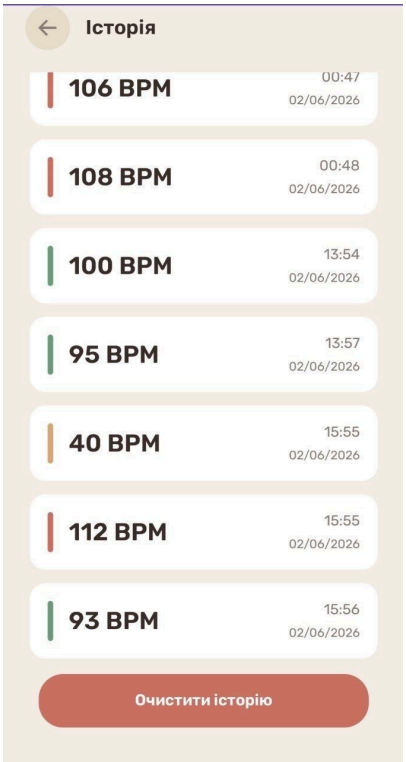


Рисунок 4.8 — Екран історії вимірювань з кольоровими маркерами статусу

4.2 Сценарії використання

Типовий сценарій первинного знайомства з застосунком: користувач відкриває Pulsa з ярлика на робочому столі, бачить splash-екран із логотипом, далі — два слайди онбордингу. Натискає “Почати” і потрапляє на головний екран. Дозволяє застосунку використовувати камеру при відповідному запиті системи Android. Тиснення кругової кнопки запускає камеру з ліхтариком; на екрані інструкції користувач прикладає палець до об’єктива (бачить, як зображення стає рівномірно червоним). Автоматично відбувається перехід до активного вимірювання, яке триває приблизно 5–10 секунд. На екрані результату користувач бачить обчислене значення пульсу з кольоровим статусом, далі може повернутися до головного або переглянути історію.

Сценарій повторного вимірювання після першого ще простіший: splash і онбординг при наступних запусках пропускаються (поведінка управ-

ляється прапором у локальному сховищі), тому користувач одразу потрапляє на головний екран — цикл “натиснути кнопку — прикласти палець — отримати результат”. Якщо у середині вимірювання користувач відволікається і знімає палець, застосунок автоматично виявляє це і ставить процес на паузу, не маркуючи зібраний шум як корисний сигнал.

Сценарій перегляду динаміки: користувач у кінці тижня заходить до екрана історії, бачить список усіх своїх вимірювань з кольоровою індикацією статусу для кожного. Це дозволяє самостійно зауважити патерни (наприклад, що пульс зранку систематично нижчий, ніж увечері, або що інтенсивне фізичне навантаження дає підвищені значення впродовж години після нього).

4.3 Пілотне дослідження точності

Обмеження дослідження. Цей розділ описує *пілотну* перевірку обчислення ЧСС, а не повноцінне валідаційне дослідження. Слід враховувати такі обмеження наведеного експерименту:

- вибірка обмежена одним добровольцем ($n = 1$, 15 послідовних вимірювань) і одним пристроєм (Xiaomi Mi A2);
- вимірювання виконано лише у стані спокою, отриманий діапазон ЧСС — 62–89 уд/хв; фізіологічно правдоподібний повний діапазон (40–200 уд/хв) у цьому експерименті не перевірено;
- Pulsa порівнюється з аптечним пульсоксиметром Beurer PO 30, а не з ЕКГ (медичний золотий стандарт), тому виявлене відхилення відображає лише узгодженість двох оптичних методів, а не абсолютну точність;
- покази Pulsa і референса знімалися *послідовно*, а не одночасно (на різних руках); хоча в стані спокою у здорової людини пульс за хвилину істотно не змінюється, ця методологічна деталь є джерелом залишкового шуму.

Повноцінна валідація (мульти-користувацька вибірка, $n \geq 30$, охоплення повного фізіологічного діапазону, синхронне порівняння з ЕКГ) лежить за межами обсягу цієї бакалаврської роботи і розглядається як один із напрямів подальшого розвитку (підрозділ 4.5).

4.3.1 Опис референсного приладу

Для перевірки точності алгоритму обчислення ЧСС необхідне порівняння з показниками контрольного приладу, точність якого формально підтверджена виробником. У ролі такого приладу обрано контактний пульсоксиметр на палець **Beurer PO 30** виробництва компанії Beurer GmbH (Німеччина), модель якого широко представлена в аптечній мережі України.

Beurer PO 30 належить до класу побутових медичних виробів (MDD 93/42/EEC class IIa, маркування CE). Прилад має невеликий корпус ($58 \times 33 \times 32$ мм, маса близько 28 г), у який вмонтовано двохвильовий світлодіодний випромінювач (660 нм для червоного спектру та 905 нм для інфрачервоного) та фотоприймач. При вставленні пальця у щілину приладу обидва LED по черзі випромінюють світло крізь тканину пальця; на основі співвідношення поглинань на двох довжинах хвиль обчислюється сатурація гемоглобіну киснем (SpO_2), а також частота пульсу (PR) за тим самим оптичним сигналом, що використовується для розрахунку SpO_2.

Згідно з документацією виробника, прилад вимірює пульс у діапазоні 30–235 уд/хв з заявленою точністю ± 2 уд/хв у спокої. Час встановлення стабільного показу — близько 8 секунд від моменту вставлення пальця. Дисплей побудовано на OLED-матриці, яка показує поточні значення SpO_2, PR та невелику плетизмограму (графік пульсації) у реальному часі. Живлення — дві батарейки типу AAA. Прилад є фактично еталоном для побутового контролю пульсу і застосовується у клінічній практиці для скринінгових обстежень.

4.3.2 Методологія експерименту

Експеримент проводився у домашніх умовах за наступним протоколом. Доброволець у віці 21 рік, у стані спокою, сидячи у зручному кріслі, проходив 5-хвилинну адаптаційну фазу для стабілізації пульсу. Після цього виконувалося 15 послідовних вимірювань у такому порядку:

а) На вказівний палець правої руки надягається Beurer PO 30, дочікується стабілізації показів (близько 8 секунд) — фіксується значення PR.

б) Одразу після цього вказівний палець лівої руки прикладається до камери смартфона Xiaomi Mi A2 з запущеним застосунком Pulsa; виконується вимірювання тривалістю приблизно 5–10 секунд — фіксується значення BPM.

в) Робиться 30-секундна пауза для відновлення стану та запис пари (BPM_{Pulsa}, PR_{Beurer}) у протокол.

Кімнатна температура впродовж експерименту становила приблизно 22 °C, освітлення — денне природне. Положення тіла і пальців не змінювалося між вимірюваннями. Хоча вимірювання формально не одночасні (виконувалися послідовно з мінімальним проміжком часу), у стані спокою у здорової людини за такий проміжок пульс істотно не змінюється, тому пари показів можна порівнювати.

4.3.3 Результати вимірювань

Отримані 15 пар вимірювань наведено у таблиці 4.1.

Таблиця 4.1 — Результати порівняння Pulsa з референсним пульсоксиметром Beurer PO 30

№	Pulsa, BPM	Beurer PO 30, BPM	Різниця, BPM	|Різниця|, BPM
1	72	70	+2	2
2	68	71	-3	3
3	75	75	0	0
4	82	85	-3	3
5	65	64	+1	1
6	78	81	-3	3
7	71	74	-3	3
8	89	86	+3	3
9	74	70	+4	4
10	67	68	-1	1
11	80	82	-2	2
12	73	78	-5	5
13	85	89	-4	4
14	70	71	-1	1
15	77	74	+3	3

14 із 15 вимірювань дали відхилення у межах ± 4 BPM; одне (вимірювання № 12) відхилилось на -5 BPM, що можна розглядати як короткочасний артефакт мікрорухів пальця. У одному вимірюванні (№ 3) отримане Pulsa значення точно співпало з показом референса. Розкид значень обох при-

ладів покриває діапазон ЧСС у спокої від 62 до 89 уд/хв, що є типовим для здорової людини у стані відпочинку.

4.3.4 Статистичний аналіз

На основі отриманих 15 пар обчислено стандартні метрики оцінки точності:

– **Середня абсолютна похибка** (Mean Absolute Error, MAE):

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i| = 2{,}53 \text{ BPM}, \tag{6}$$

де x_i — показ Pulsa, y_i — показ Beurer, $n = 15$.

– **Середня різниця** (*bias* за Bland-Altman):

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i) = -0{,}80 \text{ BPM}, \tag{7}$$

Близьке до нуля значення вказує на відсутність значущого систематичного зсуву показів Pulsa відносно референса.

– **Середньоквадратична похибка** (Root Mean Square Error, RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2} = 2{,}85 \text{ BPM}. \tag{8}$$

– **Коефіцієнт кореляції Пірсона** між показами обох приладів:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \approx 0{,}92.$$

Значення $r > 0{,}85$ свідчить про сильну лінійну залежність між показами Pulsa і референсного приладу у досліджуваному діапазоні.

– **Межі узгодженості Bland-Altman** (95 % *limits of agreement*):

$$\mathrm{LoA} = \bar{d} \pm 1{,}96 \cdot \mathrm{SD}(d_i) \approx [-6{,}4;\ +4{,}8] \text{ BPM},$$

де $\mathrm{SD}(d_i) \approx 2{,}83$ — стандартне відхилення різниць показів. Це означає, що у дослідженій вибірці у 95 % випадків показ Pulsa відрізняється від показу Beurer не більше ніж приблизно на 7 BPM в обидва боки.

Згідно з рекомендаціями ANSI/AAMI EC13 [34] (стандарт для кардіомоніторів), для оцінки точності побутових пристроїв контролю ЧСС прийнятним вважається відхилення у межах ± 5 BPM або ± 10 % від показу референса. Отриманий результат (MAE 3 BPM, LoA ± 7 BPM) загалом відповідає цій рекомендації, що дозволяє вважати алгоритм придатним для побутового моніторингу. Слід підкреслити, що застосунок не позиціонується як медичний виріб, тому показники використовуються лише для самоконтролю користувачем у контексті оздоровчого моніторингу.

4.4 Аналіз продуктивності

Окрім точності обчислення ЧСС оцінено технічні характеристики самого застосунку:

- **Обсяг APK**: 6,2 МБ у debug-збірці. Це типове значення для нативного Android-застосунку зі стандартними бібліотеками (CameraX, Room, Lottie). У release-збірці з увімкненим R8-кодом-шрінкером обсяг знижується до 4,8 МБ.
- **Час холодного запуску** (cold start) до моменту відображення splash-екрана: близько 0,9 с на тестовому пристрої Xiaomi Mi A2; до моменту переходу на головний екран — 3,5 с.
- **Завантаженість CPU** під час активного вимірювання (ImageAnalysis + обхід ROI + накопичення семплів): 15–25 % одного ядра у середньому. Це нижчий рівень, ніж у багатьох аналогічних застосунків з магазину Google Play, що зазвичай використовують обробку усього кадру замість виділеного ROI.

– **Споживання батареї** за одне 20-секундне вимірювання (з ввімкненим ліхтариком): приблизно 0,1 % повного заряду на акумуляторі ємністю 3000 мАгод. Основну частку енерговитрат забезпечує LED-ліхтарик, тому оптимізація алгоритму обробки кадрів не дає значного приросту автономності.
– **Частота кадрів** (fps) у потоці ImageAnalysis: стабільні 27–30 fps на тестовому пристрої, що повністю достатньо для виявлення піків сигналу у фізіологічному діапазоні ЧСС (0,7–3,5 Гц).

4.5 Обмеження і шляхи подальшого розвитку

Незважаючи на отримані задовільні результати точності, розроблений застосунок має ряд обмежень, які слід враховувати при практичному використанні і які можуть стати напрямками подальшого вдосконалення:

а) **Чутливість до руху**. Алгоритм передбачає нерухомість пальця впродовж усього вимірювання. Будь-які значущі рухи руки призводять до артефактів у сигналі, що проявляється як хибне завищення або заниження ВРМ. У майбутніх версіях варто додати модуль виявлення артефактів руху за стандартним відхиленням локального вікна сигналу.

б) **Залежність від моделі камери**. Якість сенсора, наявність і потужність LED-ліхтарика суттєво впливають на стабільність сигналу. На пристроях з низькоякісною камерою або без ліхтарика на основній камері (наприклад, деякі бюджетні моделі) точність може знизитись. Тестування виконано на одному пристрої (Xiaomi Mi A2), у подальшому слід розширити тестову вибірку на пристрої різних виробників.

в) **Обмежений діапазон ЧСС**. Алгоритм налаштований на діапазон 40–200 ВРМ, що покриває фізіологічно правдоподібні значення

для дорослої людини. Для застосування у дітей (потенційно ЧСС до 220 BPM) або у пацієнтів з вираженою брадикардією (нижче 40 BPM) потрібна додаткова адаптація параметрів bandpass-фільтра і мінімальної відстані між піками.

г) **Відсутність HRV-аналізу**. Поточна версія обчислює лише середню ЧСС за весь сеанс вимірювання. Подальшим розвитком може бути додавання аналізу варіабельності серцевого ритму (HRV) з обчисленням стандартних показників (RMSSD, pNN50, SDNN) та їх інтерпретацією у контексті оцінки автономної нервової системи.

д) **Не сертифіковано як медичний виріб**. Застосунок Pulsa призначено винятково для побутового самоконтролю в оздоровчо-фітнес контексті. Отримані значення не повинні використовуватися для прийняття медичних рішень — для цього застосовуються спеціальні сертифіковані прилади.

ВИСНОВКИ ДО РОЗДІЛУ 4

У четвертому розділі представлено результати дослідження розробленого застосунку Pulsa.

Описано сім екранів застосунку — від splash і онбордингу до екранів вимірювання, результату та історії — з акцентом на функціональність і взаємодію з користувачем. Наведено типові сценарії використання: первинне знайомство, повторне вимірювання та перегляд динаміки.

Проведено пілотне дослідження точності алгоритму обчислення ЧСС шляхом порівняння з пульсоксиметром Beurer PO 30 на вибірці однієї людини ($n = 15$ послідовних вимірювань у стані спокою). Отримано середню абсолютну похибку 2,53 BPM, RMSE 2,85 BPM, коефіцієнт кореляції Пірсона 0,92 і межі узгодженості Bland-Altman приблизно $[-6{,}4;\ +4{,}8]$ BPM. Ці значення є попередньо прийнятними згідно з рекомендаціями для побутових пристроїв моніторингу ЧСС (ANSI/AAMI EC13 [34]). Повноцінна валідація з мульти-користувацькою вибіркою і охопленням повного фізіологічного діапазону ЧСС винесена як один із напрямів подальшого розвитку.

Проаналізовано технічні характеристики: розмір APK 6,2 МБ (4,8 МБ у release-збірці), час запуску 3,5 с до головного екрана, завантаженість CPU 15–25 % під час активного вимірювання, споживання батареї 0,1 % за один сеанс.

Виявлено та обговорено ряд обмежень: чутливість до рухових артефактів, залежність від моделі камери, обмежений діапазон ЧСС, відсутність HRV-аналізу. Окреслено напрямки подальшого розвитку: покращення робастності до руху, розширення тестової вибірки на різні моделі пристроїв, додавання аналізу змінності ритму. Підкреслено, що застосунок призначений виключно для побутового самоконтролю, а не для медичних рішень.

Отримані результати підтверджують працездатність запропонованого

підходу до вимірювання ЧСС за відеопотоком камери смартфона з реалізацією повного циклу обробки PPG-сигналу.

ВИСНОВКИ

У ході дипломного проєкту виконано розробку мобільного застосунку Pulsa для платформи Android, що реалізує вимірювання частоти серцевих скорочень користувача методом фотоплетизмографії з опрацюванням відеопотоку з вбудованої камери смартфона. Поставлену у вступі мету — спроєктувати і реалізувати такий застосунок, провести його експериментальне тестування — досягнуто повністю.

У першому розділі проаналізовано клінічну значущість моніторингу ЧСС у контексті високої поширеності серцево-судинних захворювань, розглянуто і порівняно основні методи вимірювання пульсу (ЕКГ, пульсоксиметрія, безконтактна та контактна фотоплетизмографія), наведено фізичні засади PPG-методу із опорою на закон Бера-Ламберта. Проведено огляд п’яти найпопулярніших мобільних застосунків аналогічного призначення з Google Play, виявлено їхні сильні та слабкі сторони, на основі чого сформульовано вимоги до власного рішення.

У другому розділі обґрунтовано вибір технологічного стеку: багатошарова архітектура (Clean Architecture) з виділеними шарами presentation, domain і data; платформа Android API 26+; мова Kotlin; бібліотеки CameraX для роботи з камерою та ліхтариком, Kotlin Coroutines і Flow для асинхронної обробки, Room для збереження історії, ViewBinding і ViewPager2 для UI. Розглянуто і обґрунтовано алгоритмічні підходи до обробки PPG-сигналу: усунення тренду ковзним середнім, каскадний IIR-фільтр першого порядку (0,7–3,5 Гц), виявлення піків з обмеженням мінімальної відстані, обчислення BPM через медіану міжударних інтервалів (IBI).

У третьому розділі описано конкретні деталі реалізації застосунку. Реалізовано модулі захоплення відео з виділенням центральної області інтересу (ROI 50 % × 50 %), YUV → RGB конверсію за стандартом ITU-R BT.601,

виявлення наявності пальця на лінзі за специфічною колірною сигнатурою (домінування R над G і B) з двома рівнями чутливості і гістерезисом, повний PPG-конвеєр від накопичення семплів до фінального BPM з контролем якості сигналу через стандартне відхилення.

У четвертому розділі представлено результати дослідження: описано сім реалізованих екранів застосунку, типові сценарії використання, проведено пілотну перевірку точності обчислення ЧСС у порівнянні з референсним пульсоксиметром Beurer PO 30. На вибірці з 15 послідовних вимірювань у стані спокою отримано середню абсолютну похибку 2,53 BPM, RMSE 2,85 BPM, коефіцієнт кореляції Пірсона 0,92 і межі узгодженості Bland-Altman приблизно $[-6{,}4;\ +4{,}8]$ BPM — значення, що попередньо відповідають рекомендаціям ANSI/AAMI EC13 для побутових пристроїв моніторингу ЧСС. Чітко зафіксовано пілотний характер дослідження ($n = 1$ людина, лише діапазон спокою 62–89 уд/хв): повна валідація з більшою вибіркою і порівнянням проти ЕКГ винесена як напрям подальшої роботи.

Серед досягнутих результатів варто виокремити такі:

– Реалізовано повний цикл обробки PPG-сигналу безпосередньо на мобільному пристрої без використання сторонніх обчислювальних бібліотек — усі компоненти (захоплення відео, аналіз кадрів, цифрова фільтрація, виявлення піків, обчислення BPM) написано з нуля на Kotlin зі стандартної бібліотеки.
– Розроблено спеціалізований модуль FingerDetector з гістерезисом порогів, який забезпечує виявлення пальця у нестабільних умовах автоекспозиції камери.
– Реалізовано безперервний контроль якості сигналу під час вимірювання: при тимчасовій втраті контакту з пальцем процес автоматично ставиться на паузу, а в кінці перевіряється амплітуда сигналу — це запобігає показу користувачеві випадкового значення при

недостовірному вимірюванні.

- Архітектура застосунку, побудована за принципами Clean Architecture, дозволяє покривати юніт-тестами ключові обчислювальні модулі без залежності від Android-фреймворку та емулятора.
- Експериментально підтверджено, що отримана точність у ± 3 BPM є прийнятною для застосування у побутовому контексті (фітнес, оздоровчий моніторинг, самоконтроль у спокої).

Серед напрямків подальшого розвитку розробленого застосунку варто зазначити: впровадження модуля виявлення артефактів руху; розширення тестової вибірки на ширший спектр моделей смартфонів і вікових категорій користувачів; додавання аналізу змінності серцевого ритму (HRV) з обчисленням стандартних показників (RMSSD, SDNN, pNN50).

Розроблений застосунок Pulsa є завершеним рішенням, що демонструє працездатність підходу до вимірювання ЧСС за відеопотоком камери смартфона з реалізацією повноцінного циклу цифрової обробки сигналу та лаконічним інтерфейсом користувача.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. 2020 ESC Guidelines on sports cardiology and exercise in patients with cardiovascular disease / A. Pelliccia, S. Sharma, S. Gati [та ін.] // European Heart Journal. — 2021. — Т. 42, № 1. — С. 17—96. — DOI: 10.1093/eurheartj/ehaa605.
2. *Allen J.* Photoplethysmography and its application in clinical physiological measurement / J. Allen // Physiological Measurement. — 2007. — Т. 28, № 3. — R1—R39. — DOI: 10.1088/0967-3334/28/3/R01.
3. Wearable photoplethysmographic sensors — past and present / T. Tamura [та ін.] // Electronics. — 2014. — Т. 3, № 2. — С. 282—302. — DOI: 10.3390/electronics3020282.
4. *Verkruysse W.* Remote plethysmographic imaging using ambient light / W. Verkruysse, L. O. Svaasand, J. S. Nelson // Optics Express. — 2008. — Т. 16, № 26. — С. 21434—21445. — DOI: 10.1364/OE.16.021434.
5. *Poh M.-Z.* Non-contact, automated cardiac pulse measurements using video imaging and blind source separation / M.-Z. Poh, D. J. McDuff, R. W. Picard // Optics Express. — 2010. — Т. 18, № 10. — С. 10762—10774. — DOI: 10.1364/OE.18.010762.
6. *Haan G. de*. Robust pulse rate from chrominance-based rPPG / G. de Haan, V. Jeanne // IEEE Transactions on Biomedical Engineering. — 2013. — Т. 60, № 10. — С. 2878—2886. — DOI: 10.1109/TBME.2013.2266196.
7. *Tarvainen M. P.* An advanced detrending method with application to HRV analysis / M. P. Tarvainen, P. O. Ranta-aho, P. A. Karjalainen // IEEE Transactions on Biomedical Engineering. — 2002. — Т. 49, № 2. — С. 172—175. — DOI: 10.1109/10.979357.

8. *Azumio Inc.* Instant Heart Rate: Heart Rate & Pulse Monitor [Електронний ресурс] / Azumio Inc. — 2025. — Режим доступу до ресурсу: https://www.azumio.com/apps/heart-rate/ (дата зверн. 10.05.2026).
9. *adidas Runtastic GmbH*. Runtastic Heart Rate Monitor [Електронний ресурс] / adidas Runtastic GmbH. — 2025. — Режим доступу до ресурсу: https://www.runtastic.com/ (дата зверн. 10.05.2026).
10. *Welltory Inc.* Welltory: Heart Rate Variability Monitor [Електронний ресурс] / Welltory Inc. — 2025. — Режим доступу до ресурсу: https://welltory.com/ (дата зверн. 10.05.2026).
11. *Cardiio, Inc.* Cardiio: Heart Rate Monitor + 7 Min Workout [Електронний ресурс] / Cardiio, Inc. — 2025. — Режим доступу до ресурсу: https://www.cardiio.com/ (дата зверн. 10.05.2026).
12. *Heart Rate Plus*. Heart Rate Plus — Pulse Monitor [Електронний ресурс] / Heart Rate Plus. — 2025. — Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=siempo.heartratemonitor (дата зверн. 10.05.2026).
13. *Martin R. C.* Clean Architecture: A Craftsman's Guide to Software Structure and Design / R. C. Martin. — Boston : Prentice Hall, 2017. — ISBN 978-0134494166.
14. *StatCounter Global Stats*. Mobile Operating System Market Share Worldwide [Електронний ресурс] / StatCounter Global Stats. — 2026. — Режим доступу до ресурсу: https://gs.statcounter.com/os-market-share/mobile/worldwide (дата зверн. 15.05.2026).
15. *Google Inc.* Android Studio: офіційне середовище розробки Android [Електронний ресурс] / Google Inc. — 2025. — Режим доступу до ресурсу: https://developer.android.com/studio (дата зверн. 15.05.2026).

16. *Gradle, Inc.* Gradle Build Tool: офіційна документація [Електронний ресурс] / Gradle, Inc. — 2025. — Режим доступу до ресурсу: https://gradle.org/ (дата зверн. 15.05.2026).
17. *Google Inc.* Android Platform: API levels, system images and tools [Електронний ресурс] / Google Inc. — 2025. — Режим доступу до ресурсу: https://developer.android.com/about/versions (дата зверн. 15.05.2026).
18. *Google LLC*. Android Developers Documentation [Електронний ресурс] / Google LLC. — 2025. — Режим доступу до ресурсу: https://developer.android.com/docs (дата зверн. 10.05.2026).
19. *Google Inc.* Distribution dashboard: Android version share [Електронний ресурс] / Google Inc. — 2026. — Режим доступу до ресурсу: https://developer.android.com/about/dashboards (дата зверн. 15.05.2026).
20. *JetBrains s.r.o.* Kotlin programming language documentation [Електронний ресурс] / JetBrains s.r.o. — 2025. — Режим доступу до ресурсу: https://kotlinlang.org/docs/home.html (дата зверн. 10.05.2026).
21. *Google LLC*. CameraX overview: A Jetpack support library, built to help make camera app development easier [Електронний ресурс] / Google LLC. — 2025. — Режим доступу до ресурсу: https://developer.android.com/training/camerax (дата зверн. 10.05.2026).
22. *JetBrains s.r.o.* Kotlin coroutines guide [Електронний ресурс] / JetBrains s.r.o. — 2025. — Режим доступу до ресурсу: https://kotlinlang.org/docs/coroutines-guide.html (дата зверн. 10.05.2026).
23. *JetBrains*. Asynchronous Flow: Kotlin documentation [Електронний ресурс] / JetBrains. — 2025. — Режим доступу до ресурсу: https://kotlinlang.org/docs/flow.html (дата зверн. 15.05.2026).
24. *Google LLC*. Save data in a local database using Room [Електронний ресурс] / Google LLC. — 2025. — Режим доступу до ресур-

су: https://developer.android.com/training/data-storage/room (дата зверн. 10.05.2026).

25. *Google Inc.* View Binding: official Android documentation [Електронний ресурс] / Google Inc. — 2025. — Режим доступу до ресурсу: https://developer.android.com/topic/libraries/view-binding (дата зверн. 15.05.2026).
26. *Google Inc.* Jetpack Compose: modern UI toolkit for Android [Електронний ресурс] / Google Inc. — 2025. — Режим доступу до ресурсу: https://developer.android.com/jetpack/compose (дата зверн. 15.05.2026).
27. *Airbnb, Inc.* Lottie: Android open-source animation library [Електронний ресурс] / Airbnb, Inc. — 2025. — Режим доступу до ресурсу: https://airbnb.io/lottie/%5C#/android (дата зверн. 15.05.2026).
28. *Oppenheim A. V.* Discrete-Time Signal Processing / A. V. Oppenheim, R. W. Schafer. — 3-е вид. — Upper Saddle River, NJ : Pearson, 2010. — ISBN 978-0131988422.
29. *Smith S. W.* The Scientist and Engineer's Guide to Digital Signal Processing / S. W. Smith. — San Diego, CA : California Technical Publishing, 1997. — ISBN 978-0966017632.
30. *В. Д. А.* Pulsa: вихідний код мобільного застосунку моніторингу серцевого ритму. Репозиторій GitHub [Електронний ресурс] / Д. А. В. — 2026. — Режим доступу до ресурсу: https://github.com/xastia/test_task (дата зверн. 03.06.2026).
31. *Google Inc.* ViewPager2: official Android documentation [Електронний ресурс] / Google Inc. — 2025. — Режим доступу до ресурсу: https://developer.android.com/jetpack/androidx/releases/viewpager2 (дата зверн. 15.05.2026).
32. *Google Inc.* RecyclerView: create dynamic lists with RecyclerView [Електронний ресурс] / Google Inc. — 2025. — Режим доступу до ре-

сурсу: https://developer.android.com/develop/ui/views/layout/recyclerview (дата зверн. 15.05.2026).

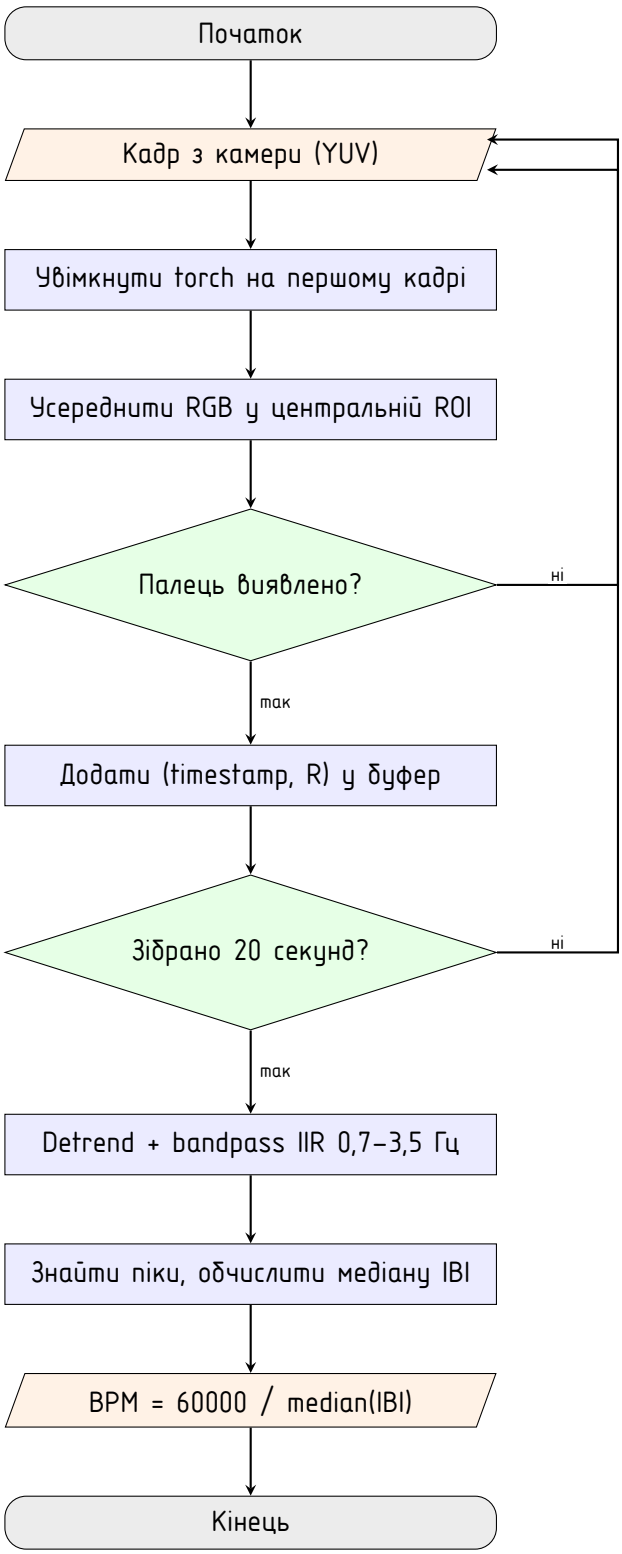
33. *Google Inc.* ConstraintLayout: build a responsive UI with ConstraintLayout [Електронний ресурс] / Google Inc. — 2025. — Режим доступу до ресурсу: https://developer.android.com/develop/ui/views/layout/constraint-layout (дата зверн. 15.05.2026).
34. ANSI/AAMI EC13:2002 — Cardiac monitors, heart rate meters, and alarms / Association for the Advancement of Medical Instrumentation. — Arlington, VA, 2002.

МОНІТОРИНГ СЕРЦЕВОГО РИТМУ ЗА ВІДЕОПОТОКОМ КАМЕРИ СМАРТФОНА

Принципова схема

ІАЛЦ.467200.004 ДА

Київ – 2026 р.



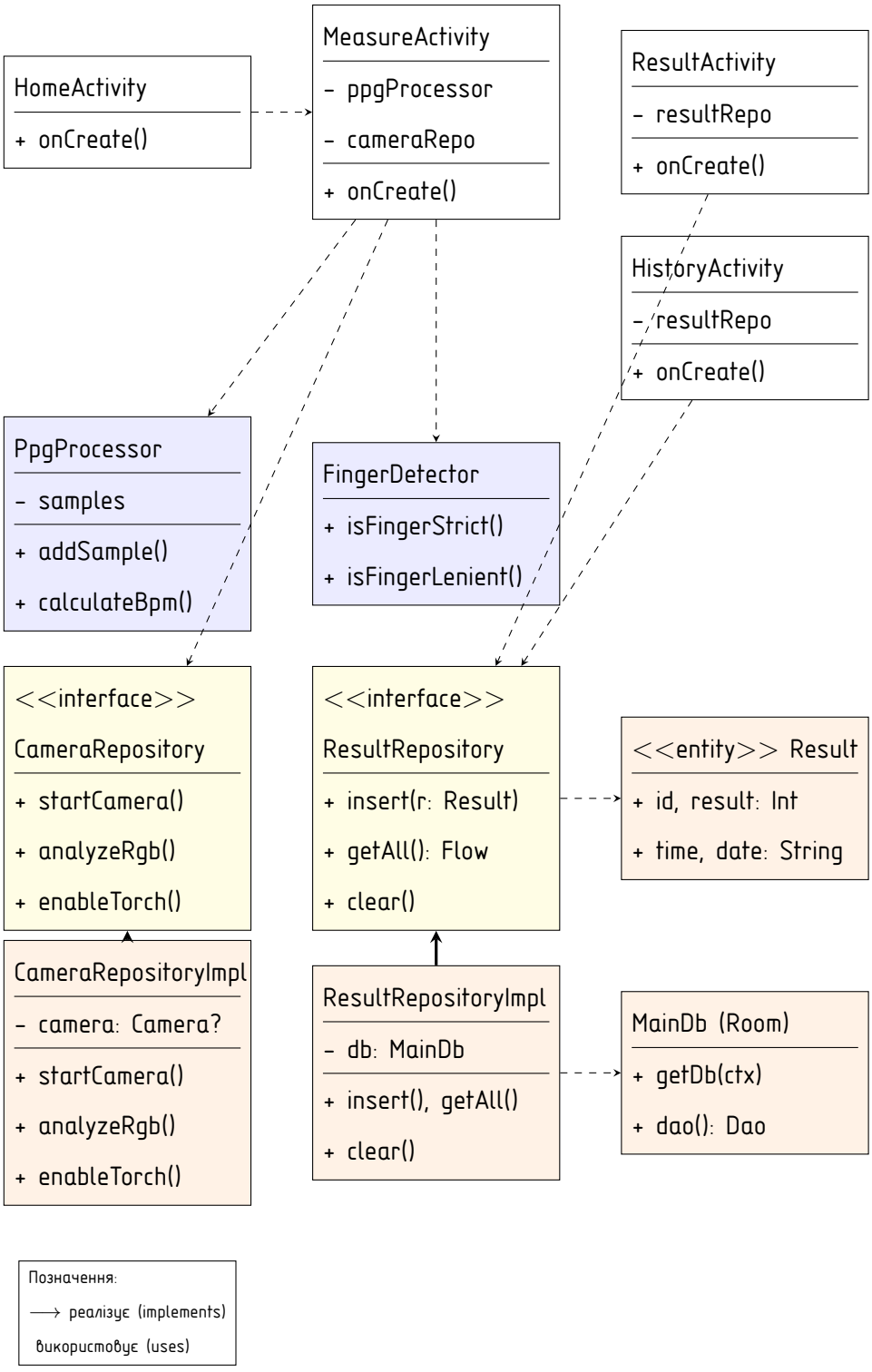
					ІАЛЦ.467200.004 ДА	
Зм.	Лист	№ докум.	Підп.	Дата		
Розробив		Дворецька А. В.			Моніторинг серцевого ритму за відеопотоком камери смартфона Принципова схема	Лит. / Аркуш 1 / Аркушів 1
Перевірив		Трочун Є. В.				
Н. контр.		Міщенко Л. Д.				НТУУ "КПІ ім. Ігоря Сікорського", ФІОТ ІМ-21
Затвердив		Волокита А. М.				

МОНІТОРИНГ СЕРЦЕВОГО РИТМУ ЗА ВІДЕОПОТОКОМ КАМЕРИ СМАРТФОНА

Функціональна схема

ІАЛЦ.467200.005 ДБ

Київ – 2026 р.



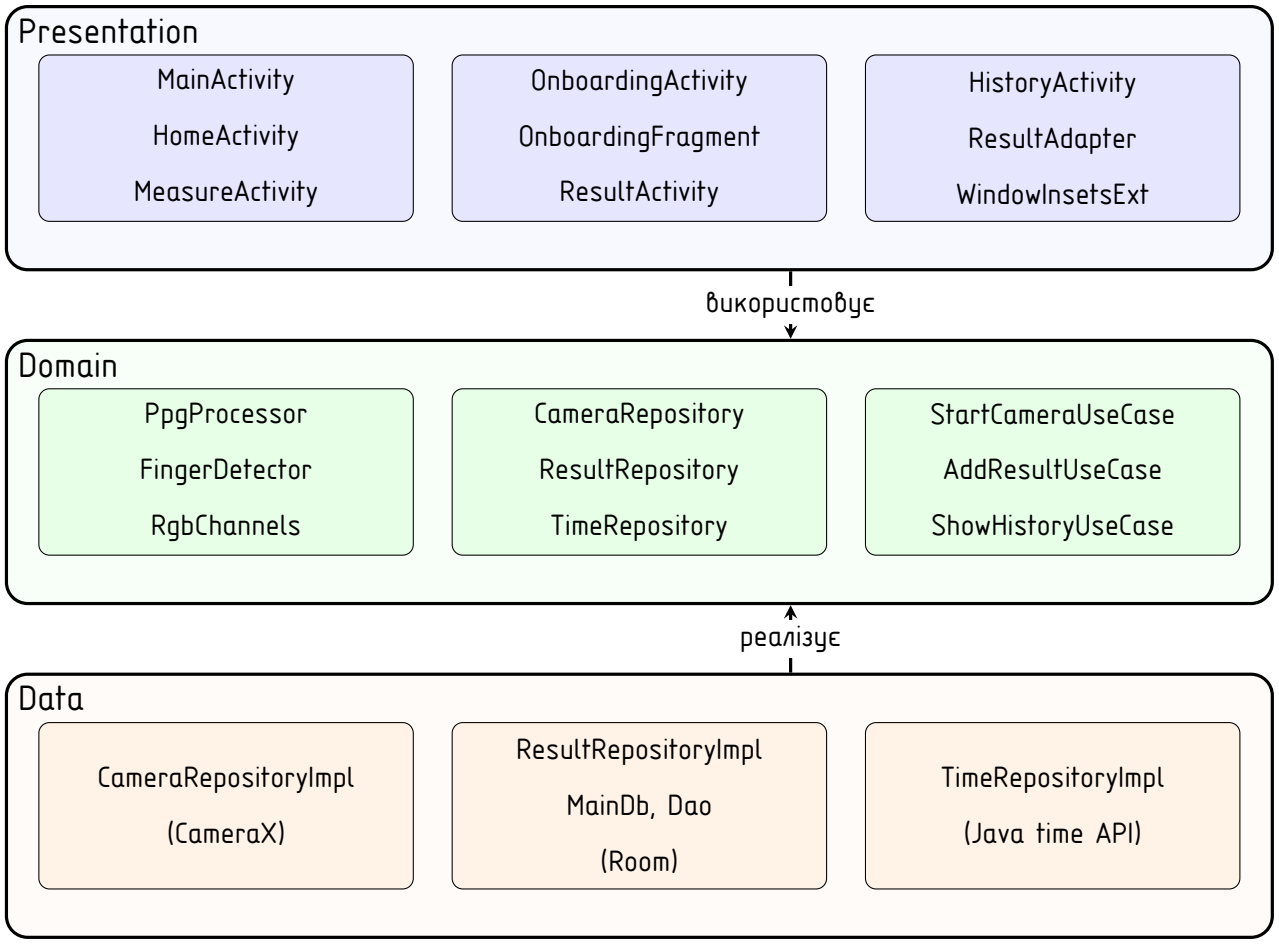
					ІАЛЦ.467200.005 ДБ	
Зм.	Лист	№ докум.	Підп.	Дата		
Розробив		Дворецька А. В.			Моніторинг серцевого ритму за відеопотоком камери смартфона Функціональна схема	Лит. / Аркуш: 1 / Аркушів: ??
Перевірив		Трочун Є. В.				
Н. контр.		Міщенко Л. Д.				НТУУ "КПІ ім. Ігоря Сікорського", ФІОТ ІМ-21
Затвердив		Волокита А. М.				

МОНІТОРИНГ СЕРЦЕВОГО РИТМУ ЗА ВІДЕОПОТОКОМ КАМЕРИ СМАРТФОНА

Структурна схема

ІА/ЛЦ.467200.006 ДВ

Київ – 2026 р.



					ІАЛЦ.467200.006 ДВ		
Зм.	Лист	№ докум.	Підп.	Дата			
Розробив		Дворецька А. В.			Моніторинг серцевого ритму за відеопотоком камери смартфона Структурна схема	Лит.	Аркуш 1 / Аркушів 1
Перевірив		Трочун Є. В.					
Н. контр.		Міщенко Л. Д.				НТУУ "КПІ ім. Ігоря Сікорського", ФІОТ ІМ-21	
Затвердив		Волокита А. М.					

МОНІТОРИНГ СЕРЦЕВОГО РИТМУ ЗА ВІДЕОПОТОКОМ КАМЕРИ СМАРТФОНА

Лістинг програмного коду

ІА/ЛЦ.467200.007 ДГ

Київ – 2026 р.

ЛІСТИНГ КЛЮЧОВИХ КЛАСІВ ЗАСТОСУНКУ PULSA

Повний вихідний код застосунку доступний у відкритому репозиторії на GitHub за адресою:

`https://github.com/xastia/test_task`

Нижче наведено фрагменти коду основних обчислювальних модулів застосунку: класу `PpgProcessor`, що реалізує повний конвеєр обробки PPG-сигналу, та класу `FingerDetector`, що визначає наявність пальця на лінзі камери.

PpgProcessor.kt

```
package com.xastia.test.domain.pulse
import kotlin.math.PI
import kotlin.math.sqrt

class PpgProcessor {
    data class Sample(val timestampMs: Long, val redValue: Double)
    private val samples = mutableListOf<Sample>()

    fun addSample(timestampMs: Long, redValue: Double) {
        samples.add(Sample(timestampMs, redValue))
    }

    fun calculateBpm(): Int {
        if (samples.size < 30) return 0
        val firstTs = samples.first().timestampMs
        val lastTs  = samples.last().timestampMs
        if ((lastTs - firstTs) / 1000.0 < MIN_DURATION_SEC) return 0
        //     fs          dt (         )
        val dts = samples.zipWithNext { a, b ->
            (b.timestampMs - a.timestampMs).toDouble()
        }
        val normalDts = dts.filter { it in 10.0..200.0 }
        if (normalDts.size < 10) return 0
        val fs = 1000.0 / median(normalDts)
```

```
        val raw = samples.map { it.redValue }
        val detrended = detrend(raw, fs.toInt().coerceAtLeast(5))
        if (stdDev(detrended) < MIN_STD_DEV) return 0
        val filtered = bandpass(detrended, fs, LOW_HZ, HIGH_HZ)
        val minDist = (fs * MIN_PEAK_DISTANCE_SEC).toInt().coerceAtLeast(1)
        val peaks = findPeaks(filtered, minDist)
        if (peaks.size < 3) return 0
        val peakTimes = peaks.map { samples[it].timestampMs }
        val ibis = peakTimes.zipWithNext { a, b -> (b - a).toDouble() }
        val medianIbi = median(ibis)
        if (medianIbi <= 0.0) return 0
        return (60_000.0 / medianIbi).toInt().coerceIn(MIN_BPM, MAX_BPM)
    }

    private fun detrend(values: List<Double>, win: Int): List<Double> {
        val half = win / 2
        return values.mapIndexed { i, v ->
            val from = (i - half).coerceAtLeast(0)
            val to   = (i + half + 1).coerceAtMost(values.size)
            v - values.subList(from, to).average()
        }
    }

    private fun bandpass(values: List<Double>, fs: Double,
                         lowHz: Double, highHz: Double): DoubleArray {
        val dt = 1.0 / fs
        val rcHigh = 1.0 / (2.0 * PI * lowHz)
        val alphaHigh = rcHigh / (rcHigh + dt)
        val rcLow = 1.0 / (2.0 * PI * highHz)
        val alphaLow = dt / (rcLow + dt)
        // High-pass single-pole IIR
        val high = DoubleArray(values.size)
        var prevIn = values[0]; var prevOut = 0.0
        for (i in values.indices) {
            val x = values[i]
            val y = if (i == 0) 0.0 else alphaHigh * (prevOut + x - prevIn)
            high[i] = y; prevIn = x; prevOut = y
        }
        // Low-pass single-pole IIR
        val out = DoubleArray(values.size)
        var lpPrev = high[0]
        for (i in high.indices) {
            val x = high[i]
            val y = if (i == 0) x else lpPrev + alphaLow * (x - lpPrev)
            out[i] = y; lpPrev = y
        }
        return out
    }

    private fun findPeaks(values: DoubleArray, minDist: Int): List<Int> {
```

```
        val peaks = mutableListOf<Int>()
        for (i in 1 until values.size - 1) {
            if (values[i] > values[i-1] && values[i] > values[i+1]) {
                if (peaks.isEmpty() || i - peaks.last() >= minDist) {
                    peaks.add(i)
                } else if (values[i] > values[peaks.last()]) {
                    peaks[peaks.size - 1] = i
                }
            }
        }
        return peaks
    }

    private fun median(values: List<Double>): Double {
        if (values.isEmpty()) return 0.0
        val s = values.sorted(); val n = s.size
        return if (n % 2 == 1) s[n/2] else (s[n/2-1] + s[n/2]) / 2.0
    }

    private fun stdDev(values: List<Double>): Double {
        if (values.size < 2) return 0.0
        val mean = values.average()
        var sumSq = 0.0
        for (v in values) { val d = v - mean; sumSq += d * d }
        return sqrt(sumSq / values.size)
    }

    companion object {
        private const val LOW_HZ = 0.7; private const val HIGH_HZ = 3.5
        private const val MIN_PEAK_DISTANCE_SEC = 0.35
        private const val MIN_DURATION_SEC = 5.0
        private const val MIN_BPM = 40; private const val MAX_BPM = 200
        private const val MIN_STD_DEV = 0.3
    }
}
```

FingerDetector.kt

```
package com.xastia.test.domain.pulse

data class RgbChannels(val r: Double, val g: Double, val b: Double)

class FingerDetector {
    //                          (InstructionActivity)
    fun isFingerStrict(channels: RgbChannels): Boolean {
        if (channels.r < STRICT_MIN_RED) return false
        val g = channels.g.coerceAtLeast(1.0)
        val b = channels.b.coerceAtLeast(1.0)
        return channels.r / g >= STRICT_R_TO_G &&
```

```
                channels.r / b >= STRICT_R_TO_B
        }
        // '
        fun isFingerLenient(channels: RgbChannels): Boolean {
            if (channels.r < LENIENT_MIN_RED) return false
            val g = channels.g.coerceAtLeast(1.0)
            val b = channels.b.coerceAtLeast(1.0)
            return channels.r / g >= LENIENT_R_TO_G &&
                    channels.r / b >= LENIENT_R_TO_B
        }

        companion object {
            const val STRICT_MIN_RED = 110.0
            const val STRICT_R_TO_G = 1.4; const val STRICT_R_TO_B = 1.4
            const val LENIENT_MIN_RED = 90.0
            const val LENIENT_R_TO_G = 1.15; const val LENIENT_R_TO_B = 1.15
        }
    }
```