Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

«Допущено до захисту»
Завідувач кафедри комп’ютерних наук, доктор технічних наук, професор
_____ Андрій БОНДАРЧУК
(підпис)
« ___ » _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня «Магістр»
Спеціальність 122 Комп’ютерні науки
Освітня програма 122.00.02 Інформаційно-аналітичні системи
Тема роботи: Система моніторингу та прогнозування стану здоров’я людини

Виконав
студент групи ІАСм-1-24-1.4д
Білан Максим Павлович
(ПІБ)
_____
(підпис)

Науковий керівник
доктор технічних наук, професор
(науковий ступінь, вчене звання)
_____ Олександр БУШМА
(підпис)

Київ – 2025

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

«Затверджую»
Завідувач кафедри комп'ютерних наук,
кандидат технічних наук, доцент
_____ Ірина МАШКІНА
(підпис)

ЗАВДАННЯ НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

студенту групи ІАСм-1-24-1.4д

Білану Максиму Павловичу

Тема роботи «Система моніторингу та прогнозування стану здоров'я людини»

1. Вихідні дані: *Сучасні методи аналізу фізіологічних сигналів, статистичні алгоритми виявлення аномалій, публікації у сфері цифрової медицини та персоналізованого моніторингу*
2. Основні завдання: *Здійснити огляд наукових робіт та існуючих рішень у сфері моніторингу стану здоров'я. Розробити структуру та зміст магістерської роботи. Обрати й обґрунтувати методи аналізу та прогнозування фізіологічних даних. Здійснити збір, попередню обробку та очищення даних. Розробити алгоритми виявлення аномалій на основі статистичних методів та методів машинного навчання. Реалізувати модуль прогнозування короткострокових змін стану організму. Провести оцінювання ефективності застосованих підходів.*
3. Пояснювальна записка: *Обсяг – до 70 стор. формату А4 комп'ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.*
4. Графічні матеріали: *презентація*.
5. Додатки: *лістинги програм*
6. Строк подання роботи на кафедру: *1.12.2025.*

Науковий керівник
д.т.н., професор
_____ О. В. Бушма

Завдання прийняв до виконання:
« 1 » грудня 2024 р.
_____ М. П. Білан

ІНДИВІДУАЛЬНИЙ ПЛАН
ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ
студента групи ІАСм-1-24-1.4д
Білана Максима Павловича
Тема роботи «Система моніторингу та прогнозування стану здоров'я людини»

№	Назва етапу дипломної роботи	Строк виконання етапу роботи	Примітка
1	Формування теми дипломної роботи магістра	08.11.2024	Виконано
2	Ознайомлення з методичними рекомендаціями до дипломної роботи	18.11.2024	Виконано
3	Складання змісту та календарного плану роботи	01.12.2024	Виконано
4	Підбір літератури, складання завдання	14.12.2024	Виконано
5	Написання реферату та вступу	28.01.2025	Виконано
6	Написання першого розділу	07.06.2025	Виконано
7	Написання другого розділу	09.07.2025	Виконано
8	Написання третього розділу	22.09.2025	Виконано
9	Написання висновків	30.09.2025	Виконано
10	Виправлення зауважень	30.10.2025	Виконано
11	Створення презентації	11.11.2025	Виконано
12	Захист матеріалів дипломної роботи на засіданні кафедри	18.11.2025	Виконано
13	Офіційний захист матеріалів дипломної роботи на засіданні екзаменаційної комісії	18.12.2025	

Здобувач Білан М.П. _____
Керівник роботи Бушма О.В. _____

Анотація кваліфікаційної роботи

Актуальність: Сучасні носимі сенсорні пристрої збирають великі обсяги фізіологічних даних у реальному часі, проте більшість із них обмежуються лише базовим відображенням показників. Своєчасне виявлення відхилень у стані здоров’я є критично важливим для запобігання ускладненням та забезпечення безпеки користувача. Використання сучасних методів аналізу даних і алгоритмів машинного навчання дозволяє створити персоналізовану систему оцінки та прогнозування змін фізіологічного стану, що сприятиме підвищенню якості моніторингу та ранньому виявленню потенційних ризиків.

Мета: Розробити систему персоналізованого моніторингу та прогнозування стану здоров’я, яка на основі історичних фізіологічних даних зможе виявляти відхилення та передбачати найближчі зміни показників, забезпечуючи підвищення ефективності контролю здоров’я користувача.

Об’єкт дослідження: Процес моніторингу та прогнозування фізіологічних показників людини на основі методів аналізу даних та машинного навчання.

Предмет дослідження: Методи аналізу даних та алгоритми машинного навчання, застосовані для прогнозування змін фізіологічних показників та оцінки стану здоров’я на основі історичних даних користувача.

Завдання роботи:

- Здійснити пошук, відбір та аналіз сучасних рішень і джерел даних у сфері цифрового моніторингу стану здоров’я.
- Обрати та обґрунтувати методи аналізу даних і алгоритми машинного навчання для виявлення відхилень та прогнозування змін фізіологічних показників.
- Реалізувати програмний прототип системи SmartHealth для збору, збереження, аналізу та візуалізації фізіологічних даних користувача.

- Провести тестування системи, оцінити коректність роботи аналітичних модулів та якість прогнозування стану здоров’я.

Методи дослідження: Аналіз наукових джерел та сучасних рішень у сфері цифрової медицини для виявлення актуальних підходів до моніторингу та прогнозування стану здоров’я; методи математичної статистики для обробки фізіологічних сигналів; застосування алгоритмів машинного навчання для виявлення аномалій та прогнозування показників; системний аналіз для проєктування архітектури програмної системи та її аналітичного ядра.

Наукова новизна дослідження: Вперше запропоновано підхід до персоналізованого моніторингу стану здоров’я, що поєднує класичні статистичні методи та алгоритми машинного навчання в єдиному аналітичному ядрі. Удосконалено методику виявлення аномалій у фізіологічних сигналах шляхом інтеграції індивідуальних динамічних норм з моделлю Isolation Forest, навченою на ковзних вікнах. Запропоновано підхід до короткострокового прогнозування фізіологічних показників у реальному часі на основі експоненціального згладжування, що дозволяє оцінювати ризики ще до появи критичних відхилень.

Практичне значення дослідження: Розроблена система SmartHealth забезпечує автоматизований повний цикл роботи з фізіологічними даними - від отримання сигналів у реальному часі до їх очищення, аналізу та формування персоналізованих висновків про стан користувача. Інтеграція статистичних методів та алгоритмів машинного навчання дозволяє підвищити точність виявлення аномальних станів та оперативно реагувати на потенційні зміни у фізіологічних показниках. Реалізований механізм короткострокового прогнозування дає змогу оцінювати можливі ризики погіршення стану здоров’я, що підвищує безпеку користувача та може бути використано у медичних, спортивних і побутових умовах.

Ключові слова: моніторинг стану здоров’я, фізіологічні сигнали, машинне навчання, аномалії, прогнозування, Z-score, Median Absolute Deviation, Mahalanobis Distance, Isolation Forest, носимі пристрої, обробка даних, персоналізація моделей.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ML (Machine Learning) - машинне навчання, галузь штучного інтелекту, що вивчає методи побудови моделей на основі даних.

ШІ - штучний інтелект.

AI-lite - спрощений режим алгоритмічної аналітики без повноцінних моделей машинного навчання, з використанням статистичних методів.

IF (Isolation Forest) - алгоритм машинного навчання для виявлення аномалій у багатовимірних даних.

HR (Heart Rate) - частота серцевих скорочень (ударів за хвилину).

SpO_2 - сатурація крові киснем (%).

Temp - температура тіла (°C).

HRV (Heart Rate Variability) - варіабельність серцевого ритму.

API (Application Programming Interface) - програмний інтерфейс для взаємодії між компонентами системи.

БД - база даних.

UI (User Interface) - користувацький інтерфейс.

PPG (Photoplethysmography) - фотоплетизмографія, оптичний метод вимірювання пульсу та SpO_2.

ECG (Electrocardiography) - електрокардіографія, метод реєстрації електричної активності серця.

CSV (Comma-Separated Values) - текстовий формат зберігання табличних даних.

СМСЗ - Системи моніторингу стану здоров’я

Вступ

У сучасних умовах зростаючого навантаження на систему охорони здоров’я, збільшення кількості хронічних захворювань, підвищення рівня стресу та численних факторів ризику особливого значення набувають інструменти безперервного та персоналізованого моніторингу стану здоров’я. Традиційні підходи, що передбачають періодичні медичні огляди, не забезпечують раннього виявлення відхилень та не дозволяють своєчасно реагувати на потенційні загрози.

Паралельно активно розвиваються цифрові технології, мобільні сенсори та методи штучного інтелекту, які дають змогу здійснювати збір, аналіз та інтерпретацію фізіологічних сигналів у реальному часі. Пристрої моніторингу, датчики споживчого та медичного рівня, системи обробки біомедичних даних і алгоритмічні методи виявлення аномалій стали основою нового напряму - персоналізованої превентивної медицини.

Науковці та розробники приділяють значну увагу цій тематиці. Різні аспекти збору та аналізу фізіологічних сигналів, побудови інтелектуальних систем підтримки прийняття рішень, а також використання алгоритмів машинного навчання в задачах медичного моніторингу розглянуті в роботах Barua S., Clifton D., Tamura T., Chen W., та багатьох інших дослідників. Окремий інтерес викликають питання автоматичного виявлення відхилень життєвих показників, оцінки ризиків та побудови персоналізованих моделей здоров’я.

Попри суттєві успіхи, більшість існуючих рішень має низку обмежень - залежність від хмарних сервісів, відсутність персоналізованих алгоритмів, вузькі можливості інтерпретованості, складність адаптації під різні сценарії використання та недостатня гнучкість при роботі з нестабільними або шумними даними. Це актуалізує потребу у створенні нових інтелектуальних систем, здатних працювати автономно, детектувати аномалії на основі індивідуальних норм користувача, інтерпретувати рішення моделі зрозумілою мовою та функціонувати у режимі реального часу.

У межах цього дослідження створено систему SmartHealth - програмно-аналітичну платформу для моніторингу стану здоров’я, яка поєднує інструменти збору фізіологічних сигналів, алгоритми персоналізованого аналізу та модулі виявлення аномалій. На відміну від багатьох існуючих рішень, система реалізує комбіновану модель оцінки стану, що включає статистичні методи, індикатори відхилень (z-score), жорсткі клінічні правила та машинне навчання (Isolation Forest), а також забезпечує пояснюваність прийнятих рішень.

Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати науково-технічні підходи до цифрового моніторингу здоров’я та обробки фізіологічних сигналів;

– узагальнити принципи роботи сучасних сенсорних технологій та методів аналізу;

– дослідити застосування алгоритмів штучного інтелекту у виявленні аномалій;

– сформувати вимоги до програмної системи моніторингу;

– обрати модель розробки та архітектурний підхід;

– реалізувати програмну систему SmartHealth, включно з базою даних, алгоритмічним ядром та вебінтерфейсом;

– розробити механізми персоналізованого аналізу, z-score оцінки, Explainable AI та моделей машинного навчання;

– виконати тестування, валідацію та оцінку точності роботи системи;

– сформулювати рекомендації щодо впровадження та можливостей подальшого розвитку.

Об’єкт дослідження - інтелектуальні системи моніторингу фізіологічних показників.

Предмет дослідження - методи збору, обробки та інтерпретації фізіологічних сигналів, а також алгоритми виявлення аномалій у даних про стан здоров’я.

Практичне значення роботи полягає у створенні функціональної платформи персоналізованого медичного моніторингу, що може використовуватися у побутових умовах, у сфері телемедицини, у системах

раннього виявлення загроз здоров’ю, у спортивній медицині та у хронічних пацієнтів для самоконтролю. Результати роботи можуть бути основою для подальших досліджень у напрямі інтеграції біомедичних сенсорів, прогнозних моделей та мультифакторного оцінювання стану пацієнта.

РОЗДІЛ 1. ПОНЯТТЯ СИСТЕМ МОНІТОРИНГУ СТАНУ ЗДОРОВ'Я ЛЮДИНИ

1.1. Актуальність тематики та стан проблеми

Забезпечення безперервного контролю за станом здоров'я людини - один з центральних викликів сучасної медицини, що перебудовується від реактивної моделі («лікувати подію») до превентивної («попереджати подію»). Зростання поширеності неінфекційних захворювань, постковідних ускладнень, розладів, пов'язаних зі стресом та способом життя, потребує переходу від епізодичних оглядів до моніторингу у реальному часі, коли зміни фізіологічних показників фіксуються завчасно, а користувач отримує зрозумілі попередження та рекомендації.

Системи моніторингу стану здоров'я- це апаратно-програмні комплекси, які збирають, передають, зберігають, аналізують і візуалізують біомедичні дані (пульс, сатурація, температура, артеріальний тиск, частота дихання, варіабельність серцевого ритму, активність, сон тощо), аби оперативно виявляти відхилення[17]. У найзагальнішому вигляді СМСЗ складаються з: (1) сенсорів та джерел даних; (2) каналу передачі (BLE/Wi-Fi/MQTT/мобільний зв'язок); (3) підсистеми зберігання (локальна БД або хмара); (4) аналітичного ядра (правила, статистика, машинне навчання/ШІ[2]); (5) інтерфейсу користувача та сповіщень; (6) сервісів інтеграції (API, обмін з ЕМЗ/EHR)[13].

Поняття та класифікація СМСЗ

СМСЗ класифікують за кількома осями.

1. За місцем застосування:

- *Клінічні*: стаціонарні монітори, телеметрія у відділеннях інтенсивної терапії, кардіореанімації. Висока точність і сертифікація, але обмежена мобільність і висока вартість.
- *Домашні*: побутові тонометри, пульсоксиметри, термометри з передачею у застосунок.

- *Носимі (wearables)*: смартгодинники/браслети з PPG/акселерометрами/термодатчиками; перевага - безперервність і пасивний збір.

2. За призначенням:

- *Профілактичні* (раннє виявлення ризиків), *діагностичні* (супровід обстеження/лікування), *реабілітаційні* (контроль відновлення), *фітнес-орієнтовані* (контроль навантаження та способу життя).

3. За архітектурою обчислень:

- *Локальні (edge)* - аналіз на пристрої/комп'ютері користувача; низькі затримки, вища приватність.

- *Хмарні* - централізована обробка; легша масштабованість і спільний доступ лікаря/пацієнта.

- *Гібридні* - комбінація локальної обробки з хмарним зберіганням або навпаки.

4. За способом аналізу:

- *Порогові/правила* (простота, пояснюваність).

- *Статистичні моделі* (ковзні середні, z-огляди, робастні метрики).

- *МЛ/ШІ-підходи* (виявлення аномалій, класифікація, прогнозування, пояснюваний ШІ).

5. За ступенем персоналізації:

- *Універсальні* (порівняння з «популяційною нормою»).

- *Персоналізовані* (базова лінія та моделі під конкретного користувача; адаптація з часом).

1.2. Загальна характеристика сучасних цифрових медичних систем

Сучасні цифрові медичні системи є результатом інтеграції досягнень біомедицини, інформаційних технологій, мікроелектроніки та штучного інтелекту. Вони забезпечують перехід від фрагментарного спостереження до безперервного, контекстно-залежного моніторингу фізіологічних показників людини у реальному часі. Основна мета таких систем полягає у підвищенні ефективності профілактики, ранньої діагностики та підтримки лікування шляхом автоматизації збору, аналізу та інтерпретації біомедичних даних[3].

Цифрові медичні системи (ЦМС) охоплюють широкий спектр технологій - від простих мобільних додатків для контролю фізичної активності до комплексних телемедичних платформ, що інтегрують лікарів, пацієнтів, лабораторії, страхові служби та аналітичні сервіси. Умовно їх можна розділити на три категорії: клінічні, споживчі та гібридні рішення.

Клінічні системи орієнтовані на медичні установи та використовуються для спостереження за пацієнтами у стаціонарах і під час лікування вдома. Вони суворо регламентуються стандартами безпеки, валідації та сумісності (HL7, DICOM, ISO/IEC 80001). Споживчі системи (Apple Health, Google Fit, Fitbit, Huawei Health)[20] є більш доступними, проте орієнтуються переважно на профілактичні цілі та не потребують сертифікації як медичні вироби. Гібридні системи поєднують обидва підходи, забезпечуючи передавання даних з носимих сенсорів до лікарських платформ або електронних медичних записів.

Розвиток цифрової медицини зумовлений низкою чинників: мініатюризацією сенсорів, розповсюдженням мобільного Інтернету, удосконаленням енергозберігаючих процесорів, розширенням можливостей хмарних обчислень і, найголовніше, впровадженням алгоритмів штучного інтелекту[1]. Ці зміни зробили можливим створення систем, які не просто фіксують дані, а здатні робити висновки та рекомендації на їх основі.

Архітектурні принципи побудови цифрових медичних систем

Типова цифрова медична система складається з кількох логічних рівнів, кожен з яких виконує власну функцію у процесі збору, обробки та подання інформації:

1. **Рівень сенсорів та збору даних.**

На цьому рівні здійснюється вимірювання фізіологічних параметрів: частоти серцевих скорочень (ЧСС), насичення крові киснем (SpO_2), температури тіла, електрокардіограми (ЕКГ), артеріального тиску, частоти дихання, активності, положення тіла. Для цього використовуються мініатюрні сенсори, розміщені у носимих пристроях або безпосередньо на шкірі. Дані оцифровуються та передаються через бездротові канали (Bluetooth, Wi-Fi, ZigBee, LTE).

2. **Рівень попередньої обробки даних.**

Тут виконуються фільтрація шумів, компенсація артефактів руху, нормалізація шкал і видалення пропусків. Часто використовуються алгоритми цифрової обробки сигналів (DSP): ковзні середні, медіанна фільтрація, вейвлет-перетворення, перетворення Фур'є. Важливою функцією є синхронізація даних з різних сенсорів за часовою міткою.

3. **Рівень аналітики та прийняття рішень.**

На цьому етапі застосовуються алгоритми статистичного аналізу або машинного навчання для виявлення відхилень, прогнозування тенденцій і оцінювання ризиків. Алгоритми можуть працювати локально (edge computing) або у хмарному середовищі (cloud AI).

4. **Рівень зберігання даних.**

Використовуються бази даних (SQLite, PostgreSQL, MongoDB, InfluxDB) для структурування часових рядів. У більш складних системах реалізовано архітектури типу data lake, що дозволяють інтегрувати сенсорні потоки з іншими медичними джерелами (електронні медичні карти, лабораторні звіти, діагностичні зображення).

5. **Рівень візуалізації та взаємодії.**

Забезпечує відображення показників у вигляді графіків, кольорових індикаторів і звітів. Окремі системи надають рекомендації або автоматичні попередження (alert) на основі аналітичних правил чи моделей штучного інтелекту.

Основні функціональні можливості сучасних цифрових систем

Сучасні цифрові медичні системи виконують такі ключові функції:

- безперервний моніторинг життєвих показників у режимі реального часу;
- виявлення відхилень та подання користувачу попереджень;
- ведення історії спостережень і побудова динаміки змін;
- обчислення похідних параметрів (варіабельність серцевого ритму, середня температура тіла, час відновлення після навантаження);
- автоматичне визначення типів активності (сон, ходьба, біг, відпочинок);
- інтеграція з мобільними додатками, фітнес- або медичними платформами;
- формування звітів для лікаря або користувача;
- використання аналітичних моделей для оцінки ризику (стрес, перевтома, серцева аритмія, гіпоксія тощо).

Сенсорні технології та фізичні принципи

У більшості носимих систем застосовуються кілька типів сенсорів, що реалізують різні фізичні принципи[4]:

- **Фотоплетизмографічні (PPG)** сенсори реєструють зміни відбитого світла, спричинені пульсацією крові у капілярах, і використовуються для вимірювання пульсу та SpO_2.

- **Електрокардіографічні (ЕКГ)** сенсори фіксують електричну активність серця та дозволяють визначати ритм і можливі патології.
- **Терморезистивні та інфрачервоні** сенсори вимірюють температуру тіла.
- **Акселерометри** й **гіроскопи** визначають положення та активність людини, оцінюють рівень фізичного навантаження.
- **Барометри** та **п'єзодатчики** дозволяють відстежувати висоту, дихальні коливання або мікрорухи грудної клітки.

Комбінація цих сенсорів створює мультисенсорну систему, здатну комплексно оцінювати стан користувача[7].

Алгоритмічне забезпечення цифрових медичних систем

На рівні аналітики застосовуються три основні групи алгоритмів:

1. **Правила та евристики (rule-based systems).**

Використовуються для простих випадків, коли стан можна описати фіксованими порогами. Наприклад, $SpO_2 < 90$ % - сигнал тривоги; температура > 38,5 °C - підозра на гарячку. Перевага таких підходів у простоті, але вони не враховують індивідуальних відмінностей.

2. **Статистичні методи.**

Більш гнучкі, оскільки базуються на аналізі середніх значень і дисперсій у часових вікнах. Методи ковзного середнього, медіани, MAD (Median Absolute Deviation) дозволяють адаптуватися до змінної поведінки показників. Робастні оцінки стійкі до шумів і викидів, що особливо важливо при роботі з реальними сенсорними потоками.

3. **Методи машинного навчання (ML) та штучного інтелекту (ШІ).**

Дозволяють виявляти складні нелінійні закономірності, недоступні класичним статистичним методам. Використовуються алгоритми

класифікації, регресії, кластеризації та виявлення аномалій. Залежно від наявності розмічених даних застосовуються методи з учителем (Random Forest, SVM, Logistic Regression) або без учителя (Isolation Forest, k-Means, Mahalanobis Distance)[6].

Інтеграція та інтероперабельність

Цифрові медичні системи мають підтримувати сумісність із міжнародними протоколами обміну даними[8]. Серед основних стандартів:

- **HL7 (Health Level Seven)** - обмін структурованими медичними повідомленнями;
- **FHIR (Fast Healthcare Interoperability Resources)** - новий відкритий стандарт REST API для взаємодії систем охорони здоров'я;
- **DICOM (Digital Imaging and Communications in Medicine)** - обмін діагностичними зображеннями.

Підтримка цих стандартів забезпечує інтеграцію цифрових систем у національні eHealth-платформи та дозволяє будувати комплексні рішення, у яких домашній моніторинг доповнює клінічну практику.

Попри швидкий прогрес, впровадження цифрових медичних систем супроводжується низкою труднощів:

- обмежена точність недорогих сенсорів;
- неоднорідність даних різних виробників;
- потреба у валідації моделей на репрезентативних вибірках;
- ризики витоку конфіденційної інформації;
- низький рівень цифрової грамотності частини населення[11];
- нормативні бар'єри щодо використання ШІ в медицині.

У зв’язку з цим актуальним напрямом є створення адаптивних, персоналізованих систем, здатних працювати як у медичному, так і у побутовому середовищі, забезпечуючи при цьому високу точність, автономність і прозорість.

Аналіз сучасних систем моніторингу здоров’я

Таблиця 1. Порівняння сучасних цифрових систем моніторингу здоров’я

Назва пристрою	**Основні сенсори / джерела даних**	**Типова аналітика**	**Персоналізація моделей**	**Пояснюваність результатів**	**Офлайн-робота (edge)**	**Хмара / сховище**
Apple Watch / Apple Health	PPG пульс / SpO_2, ЕКГ, акселерометр, гіроскоп, температура шкіри, сон, активність	Виявлення аритмій, оцінка сну, навантаження, сповіщення про падіння	Часткова персоналізація трендів	Лаконічна, через графіки і текстові повідомлення	Так	iCloud / локально
Fitbit (Google)	PPG, SpO_2, акселерометр,	Аналіз сну, стресу, активності	Персональні зони, тренди	Обмежена (індикатори)	Частково	Хмара Google
Samsung Galaxy Watch / Samsung Health	PPG, SpO_2, ЕКГ, акселерометр, гіроскоп, температура, тиск	Моніторинг активності, сну, стресу	Персональні тренди	Пояснення через кольорові графіки	Частково	Хмара Samsung

Huawei Watch / Huawei Health	PPG, SpO_2, акселерометр, гіроскоп, температура	Сон, активність, навантаження, SpO_2	Персональні межі	Лаконічна	Так	Хмара Huawei
Garmin / Garmin Connect	PPG, SpO_2, акселерометр, гіроскоп, барометр, GPS	Фітнес, VO_2max, сон, тренувальні навантаження	Персоналізовані показники	Через тренувальні метрики	Так	Garmin Connect Cloud
Withings (ScanWatch, BPM, Body+)	Осцилометричний тонометр, PPG, ваги (імпеданс), SpO_2	АТ, склад тіла, серцеві показники	Індивідуальні профілі	Графічна, базова	Частково	Хмара Withings
Philips HealthSuite	ЕКГ, SpO_2, АТ, ІВЛ, дихальні сенсори, інфузійні системи	Клінічний моніторинг, аналітика протоколів	Через медичні профілі	Документована, клінічна	Так	Хмара Philips HealthSuite
Masimo SafetyNet / Medtronic CareLink	SpO_2, ЧСС, дихання, температура, ЕКГ	Безперервний дистанційний моніторинг	Через профілі пацієнтів	Протокольна	Так	Хмара виробника

Open mHealth / PhysioNet / OHDSI	Різні типи біомедичних сигналів	Формати, фреймворки для аналізу	Залежно від дослідника	Повна прозорість	Так	Локально / хмара

Аналіз сучасних систем показує, що ринок персонального моніторингу здоров’я є надзвичайно різнорідним і стрімко розвивається. Його можна умовно поділити на три основні сегменти - споживчі, медичні та дослідницькі системи, - кожен із яких має власні пріоритети, технологічні рішення й обмеження(Таблиця 1).

У споживчому секторі домінують великі виробники мобільних пристроїв - Apple, Samsung, Google (Fitbit), Huawei, Garmin. Їх рішення спрямовані передусім на користувача, який хоче стежити за фізичною активністю, сном, пульсом або рівнем стресу. Ці системи відзначаються високим рівнем інтеграції з екосистемою смартфона, естетичним інтерфейсом, стабільністю роботи і зручністю використання. Проте більшість із них не є сертифікованими медичними пристроями, а тому не гарантують діагностичної точності, а отримані результати не мають юридичної сили у клінічному середовищі. Персоналізація в таких системах зазвичай обмежена - вони застосовують усереднені популяційні моделі або прості динамічні пороги[18].

Домашні медичні пристрої (наприклад, тонометри, ваги, пульсоксиметри компаній Withings або Omron) виступають проміжною ланкою між побутовим та клінічним моніторингом. Вони мають базову сертифікацію, забезпечують вищу точність вимірювань і дозволяють інтеграцію з електронними медичними записами. Основним обмеженням залишається вузька спеціалізація (контроль одного або кількох показників) і відсутність розвиненої аналітики на рівні штучного інтелекту.

Клінічні та телемедичні системи (Philips HealthSuite, Masimo SafetyNet, Medtronic CareLink) орієнтовані на лікарів і медичні установи. Вони

відповідають стандартам безпеки HL7, FHIR, DICOM, мають високу точність і підтримують дистанційний моніторинг хронічних хворих, післяопераційних пацієнтів або осіб із підвищеним ризиком серцево-судинних подій. Їхні переваги - точність, надійність, інтеграція з лікарняними інформаційними системами; основні недоліки - висока вартість і складність розгортання, що обмежує масове застосування[5].

Дослідницькі відкриті платформи, такі як Open mHealth, PhysioNet та OHDSI, відіграють важливу роль у розвитку галузі, забезпечуючи науковців відкритими наборами даних і стандартами обміну. Вони не призначені для кінцевого користувача, проте слугують основою для створення інноваційних моделей, алгоритмів машинного навчання і тестування прототипів систем моніторингу.

Узагальнюючи результати аналізу, можна зробити такі висновки про сучасний стан ринку:

1. **Фокус на превентивній медицині.** Основний тренд полягає у переході від реактивної діагностики до попередження патологій шляхом раннього виявлення відхилень. Це стимулює розвиток носимих і домашніх пристроїв.

2. **Інтеграція сенсорів і багатоканальний збір даних.** Майже всі сучасні системи використовують комбінацію кількох сенсорів, що дозволяє підвищити точність і забезпечити контекстну оцінку стану користувача.

3. **Штучний інтелект як аналітичне ядро.** Алгоритми машинного навчання поступово стають обов’язковим компонентом будь-якої сучасної медичної системи, хоча рівень їх застосування сильно різниться: від базового трендового аналізу у фітнес-трекерах до клінічних моделей прогнозування у телемедицині.

4. **Попит на персоналізацію.** Користувачі очікують індивідуалізованих рекомендацій. Саме персоналізація є основним напрямом розвитку - перехід від усереднених норм до індивідуальних базових ліній.

5. **Відкритість і стандартизація даних.** Відкриті протоколи (FHIR, HL7) стають критично важливими для взаємодії між пристроями, лікарнями та дослідницькими центрами.

6. **Безпека та етика даних.** З огляду на зростання обсягу персональних біометричних даних питання кібербезпеки та захисту приватності стає центральним. Тенденція ринку - перехід до локальної обробки (edge AI) та мінімізації відправлення даних у хмару.

Отже, сучасний ринок систем моніторингу здоров’я характеризується швидкою еволюцією, інтеграцією сенсорних технологій і алгоритмів штучного інтелекту, а також поступовим зрушенням у бік персоналізованих і автономних рішень. Водночас зберігається розрив між споживчими та клінічними системами - перші забезпечують масовість і зручність, другі - точність і довіру, але залишаються дорогими. Це створює перспективу для нових досліджень у напрямі гібридних систем, що поєднують точність медичних рішень і доступність побутових пристроїв[10].

1.3. Науково-технічні основи аналізу фізіологічних сигналів

Фізіологічні сигнали відображають функціональний стан організму людини і є основою для будь-якої системи моніторингу здоров’я. Вони містять інформацію про діяльність серцево-судинної, дихальної, нервової та інших систем організму. Науково-технічні основи аналізу таких сигналів формувалися на перетині біомедичної інженерії, статистики, фізики, електроніки та сучасних інформаційних технологій.

У системах моніторингу ці сигнали отримуються за допомогою сенсорів, що перетворюють фізіологічні процеси у вимірювані електричні або оптичні параметри. Залежно від типу сигналу, застосовуються різні фізичні принципи: фотоплетизмографія (для вимірювання пульсу та SpO_2), осцилометрія (для артеріального тиску), терморезистивний метод (для температури тіла), електрокардіографія (для електричної активності серця), пневмотахографія (для дихальних параметрів) тощо.

Загальні характеристики фізіологічних сигналів

Фізіологічні сигнали - це часові послідовності вимірювань, які відображають функціональні процеси організму людини. Вони є відображенням діяльності внутрішніх органів та систем, таких як серцево-судинна, дихальна, нервова, ендокринна тощо. У системах моніторингу здоров’я фізіологічні сигнали виступають первинним джерелом інформації, на основі якого відбувається подальша аналітика, виявлення відхилень і формування висновків про стан користувача.

Усі фізіологічні сигнали мають спільні риси, які визначають специфіку їх технічної обробки, але водночас вони різняться за фізичною природою, структурою, масштабом і вимогами до точності. Для адекватного моніторингу стану людини необхідно враховувати ці особливості на кожному етапі - від сенсорного вимірювання до алгоритмічного аналізу.

1. Безперервність та нестаціонарність

Фізіологічні процеси у живому організмі не є сталими - вони змінюються з часом навіть у межах здорового функціонування. Серцева частота, температура, насичення крові киснем, електрична активність м’язів або мозку - усі ці показники постійно коливаються внаслідок адаптації до зовнішніх умов.

Наприклад, частота серцевих скорочень у стані спокою може бути 65–75 уд./хв, але при стресі, фізичному навантаженні або навіть емоційній реакції вона змінюється у широкому діапазоні. Подібні коливання не є патологічними,

проте створюють труднощі для аналізу, адже межі «норми» не залишаються сталими в часі.

Це означає, що більшість фізіологічних сигналів мають нестаціонарний характер - їхня статистична структура змінюється з часом. Для таких сигналів застосовуються методи адаптивної фільтрації, ковзного аналізу і часових вікон, які дозволяють оцінювати локальні зміни замість глобальних середніх.

2. Індивідуальність і варіабельність

Кожна людина має власні фізіологічні особливості, які визначають індивідуальний профіль її сигналів. Те, що для однієї особи є нормою, для іншої може вказувати на відхилення. Наприклад, у тренованої людини пульс 55 уд./хв є природним, тоді як для нетренованої він може свідчити про брадикардію.

Індивідуальні відмінності зумовлені віком, статтю, генетичними факторами, рівнем фізичної активності, харчуванням, медикаментами та навіть емоційним станом. Тому персоналізований підхід у системах моніторингу стає необхідністю: межі нормальних коливань мають встановлюватися не на основі загальних клінічних норм, а на основі базової лінії конкретного користувача.

Таку персоналізацію забезпечують алгоритми, що адаптуються на індивідуальних даних, наприклад через обчислення середніх значень за перші дні використання пристрою та подальше оновлення меж «норми» в процесі експлуатації.

3. Наявність шумів і артефактів

Проблема шумів - одна з головних у біомедичних вимірюваннях. У реальних умовах носимі сенсори піддаються впливу рухів, температурних коливань, освітлення, змін вологості шкіри або тиску ремінця. Навіть короткочасне погіршення контакту між сенсором і тілом призводить до появи артефактів, які можуть бути помилково інтерпретовані як фізіологічні зміни.

Типові приклади шумів:

- **рухові артефакти** (особливо при ходьбі або тренуваннях, коли сенсор зміщується);
- **електричні наводки** від побутових пристроїв або від струмів у тілі;
- **оптичні шуми** у фотоплетизмографічних датчиках через зміну зовнішнього освітлення;
- **фізіологічні шуми**, наприклад дихальні коливання, які впливають на ЕКГ або PPG.

Для боротьби з артефактами застосовуються різні підходи: цифрова фільтрація (низько- та високочастотна), видалення екстремальних значень, апроксимація сигналів, а також використання кількох сенсорів для взаємної перевірки достовірності вимірювань.

4. Мультипараметричність сигналів

Жоден окремий показник не може дати повної картини стану здоров’я. Наприклад, підвищений пульс може бути спричинений фізичним навантаженням, хвилюванням або лихоманкою, але лише поєднання з іншими параметрами (температурою, SpO_2, активністю) дозволяє правильно інтерпретувати ситуацію.

Саме тому сучасні системи моніторингу прагнуть до **мультисенсорності** - тобто одночасного збору кількох сигналів. Комбінація серцево-судинних, дихальних, температурних і рухових параметрів дозволяє будувати багатовимірну модель фізіологічного стану.

Мультипараметричний аналіз вимагає синхронізації сигналів у часі, адже усі вимірювання повинні відображати єдиний фізіологічний момент. Для цього використовуються часові мітки (timestamp) і методи інтерполяції, що дозволяють узгодити дані з різних сенсорів.

5. Контекстна залежність

Інтерпретація фізіологічних сигналів неможлива без урахування контексту, у якому вони були зібрані. Показники, які у стані спокою свідчать про патологію, можуть бути абсолютно нормальними при навантаженні.

Наприклад, частота пульсу 110 уд./хв при сидінні може свідчити про тахікардію, але під час швидкої ходьби - це природна реакція організму. Подібно, короткочасне зниження SpO_2 під час сну не обов'язково означає гіпоксію, якщо дихальний ритм не порушений.

Для правильного тлумачення даних системи мають враховувати додаткову інформацію - положення тіла, активність користувача, час доби, рівень стресу, фізичне навантаження. Ці дані отримують з акселерометрів, гіроскопів, барометрів або контекстних сенсорів смартфона.

Іншими словами, фізіологічний сигнал має значення лише у контексті. Сучасні алгоритми аналізу використовують підхід «context-aware analytics», коли оцінка показника виконується з урахуванням умов його отримання.

6. Сезонні, добові та середовищні впливи

Фізіологічні параметри змінюються залежно від пори року, клімату, температури довкілля, вологості та навіть атмосферного тиску. У холодну погоду периферичний кровообіг зменшується, тому SpO_2 може знижуватись; у спеку збільшується частота серцевих скорочень і потовиділення[9].

Добові ритми також відіграють важливу роль. Наприклад, артеріальний тиск і температура тіла зростають удень і знижуються вночі, а варіабельність серцевого ритму досягає максимуму під час сну. Тому при побудові аналітичних моделей доцільно враховувати часові фактори - година доби, день тижня, сезон тощо.

7. Біологічна складність і нелінійність

Організм людини - це складна динамічна система, у якій зміни одного параметра можуть викликати каскад реакцій у різних фізіологічних ланцюгах.

Ця взаємозалежність робить сигнали **нелінійними**: зміна одного показника не завжди пропорційна зміні іншого.

Наприклад, зростання пульсу на 10 % не завжди супроводжується пропорційним підвищенням артеріального тиску - результат залежить від стану судин, рівня гідратації, дихання, гормонального фону.
Такі нелінійні зв’язки складно описати звичайними лінійними моделями, тому для аналізу використовують складніші математичні та алгоритмічні підходи, зокрема кореляційно-ентропійні методи, системну динаміку або моделі машинного навчання[12].

Використання статистичних і математичних моделей у фізіологічному аналізі

У сучасних цифрових системах моніторингу здоров’я аналіз фізіологічних сигналів базується на поєднанні статистичних методів та математичного моделювання. Такі моделі дозволяють перетворювати великі масиви сенсорних вимірювань у компактні узагальнені характеристики, що відображають закономірності функціонування організму. Використання математичних методів є необхідною умовою, оскільки біологічні сигнали, як правило, мають складну, нелінійну та стохастичну природу, а тому не можуть бути повністю описані звичайними аналітичними формулами.

Основним завданням математичного моделювання є побудова опису, який дозволяє прогнозувати або оцінювати стан людини на основі спостережуваних змін параметрів. Наприклад, залежність між частотою серцевих скорочень, сатурацією і температурою може бути виражена у вигляді функціональної моделі, що дозволяє оцінювати загальний рівень фізіологічного навантаження або визначати тенденцію до погіршення самопочуття.

Початковим рівнем моделювання є статистичний аналіз, який передбачає оцінювання основних характеристик сигналу - середнього значення, дисперсії, стандартного відхилення, медіани, квантилів. Ці показники визначають базовий

стан фізіологічної системи, а також дозволяють виділяти відхилення у реальному часі. У медичних дослідженнях часто застосовується ковзна оцінка статистичних параметрів, яка забезпечує адаптацію до поточних умов. Наприклад, середнє значення частоти серцевих скорочень може оновлюватися кожні 60 секунд, щоб враховувати короткочасні коливання, не втрачаючи інформації про загальний тренд.

Важливу роль відіграють імовірнісні моделі, які розглядають сигнали як випадкові процеси. Такий підхід дозволяє враховувати природну варіабельність даних, характерну для біологічних систем. Імовірнісні розподіли - нормальний, логнормальний, експоненціальний - застосовуються для оцінки розкиду вимірювань і визначення меж, за якими значення можна вважати аномальними. Наприклад, якщо поточне значення пульсу виходить за межі двох стандартних відхилень від середнього, система може розцінити це як потенційне відхилення від норми.

Класичні статистичні методи, такі як Z-score та Median Absolute Deviation, залишаються базовими інструментами для швидкої оцінки аномалій. Вони забезпечують просту та надійну перевірку у реальному часі. Проте їх обмеження полягає у відсутності здатності враховувати кореляцію між параметрами, тому у більш складних системах використовуються багатовимірні статистичні моделі. Наприклад, методи головних компонент (PCA) або незалежних компонент (ICA) дозволяють розкладати сигнали на незалежні складові, виділяючи приховані фактори, що визначають поведінку системи.

Окрім цього, у медичних системах широко застосовуються регресійні моделі - як лінійні, так і нелінійні. Лінійна регресія дає змогу встановлювати зв’язки між різними параметрами, наприклад, оцінювати, як температура впливає на частоту пульсу або як рівень насичення киснем змінюється при підвищенні серцевого навантаження. Нелінійні регресійні підходи, наприклад, поліноміальні або логістичні моделі, дозволяють описати складніші залежності, які не мають прямолінійного характеру. Такі моделі використовуються для

прогнозування станів, коли спостерігаються поступові зміни показників, наприклад розвиток гіпоксії або лихоманки.

Важливе місце займають фільтри динамічної оцінки, які дозволяють відслідковувати зміни сигналу у часі з урахуванням шуму. Найвідомішим прикладом є фільтр Калмана, який поєднує спостереження і модель процесу, формуючи оптимальну оцінку стану системи. Він широко застосовується у біомедичних пристроях, що працюють у режимі реального часу - від моніторів серцевої діяльності до систем визначення руху пацієнта. Завдяки цьому фільтру можливо компенсувати випадкові сплески у вимірюваннях і отримати згладжений сигнал, що більш точно відображає реальну фізіологічну динаміку.

Для оцінки варіабельності та складності фізіологічних сигналів використовуються фрактальні та ентропійні моделі. Вони базуються на ідеї, що здорові системи мають певний рівень хаотичності, який свідчить про їхню адаптивність. Занадто упорядкований або надто хаотичний сигнал, навпаки, вказує на порушення регуляції. Методи багатомасштабної ентропії дозволяють вимірювати складність сигналу на різних часових рівнях і таким чином виявляти ранні ознаки патологій ще до появи клінічних симптомів.

У багатовимірних моделях аналізу фізіологічних сигналів усе частіше використовуються методи кластеризації. Вони дозволяють поділити сукупність вимірювань на групи з подібними характеристиками, що відповідають певним фізіологічним станам. Такі алгоритми, як K-means або Gaussian Mixture Models, ефективно виявляють патерни поведінки, наприклад, фази спокою, активності або стресу. Використання кластерних моделей особливо цінне у персоналізованих системах, де алгоритм може самостійно «навчатися» розрізняти нормальні та аномальні стани конкретної людини.

У системах з великим обсягом даних застосовуються також байєсівські підходи, що враховують апріорну інформацію про стан користувача. Байєсівські моделі дозволяють комбінувати поточні спостереження з історичними даними, формуючи ймовірнісну оцінку того, наскільки теперішній стан є типовим або

відхиляється від очікуваного. Такий підхід особливо корисний при неповних або шумних даних, коли алгоритм повинен робити висновки з обмеженої інформації.

Особливу роль у сучасному аналізі відіграють моделі адаптивного порогу. Вони автоматично змінюють межі нормальних значень залежно від динаміки сигналу. Наприклад, якщо середній пульс користувача поступово зростає через тренування, система не сприйматиме ці зміни як патологічні, оскільки поріг адаптується. Такі моделі реалізуються через ковзні середні, експоненційні згладжування або через машинне навчання, де алгоритм самостійно визначає оптимальні межі для кожного параметра[14].

У довготривалому моніторингу ефективними є трендові моделі, що базуються на аналізі часових рядів. Вони дозволяють прогнозувати поведінку сигналу у майбутньому, використовуючи історичні дані. Серед таких моделей можна виділити авторегресійні (AR), моделі ковзного середнього (MA) та комбіновані ARIMA, які широко застосовуються у фінансовій аналітиці, але мають успішне використання і в медицині. Наприклад, за допомогою ARIMA можна передбачати зміну рівня глюкози або тиску крові на кілька годин наперед.

Математичні моделі також забезпечують можливість симуляції фізіологічних процесів. Це особливо важливо для досліджень, де не завжди можливо експериментально спостерігати зміни в організмі людини. За допомогою моделювання створюються віртуальні сценарії функціонування серцево-судинної або дихальної системи, що дозволяє вивчати вплив різних факторів - від навантаження до фармакологічних впливів.

Інтеграція статистичних і математичних підходів створює основу для гібридних моделей, у яких комбінуються класичні методи аналізу та алгоритми машинного навчання. Такі системи здатні не лише описувати вже наявні закономірності, а й самостійно відкривати нові. Наприклад, модель може спочатку використовувати регресійний аналіз для оцінки тенденцій, а потім

застосовувати кластеризацію для виявлення нових груп станів, характерних лише для конкретного користувача.

У підсумку статистичні та математичні моделі є невід'ємною складовою сучасних систем моніторингу стану здоров'я. Вони забезпечують структурованість і наукову обґрунтованість аналізу, дозволяють перейти від простого збору даних до осмисленої інтерпретації результатів, а також слугують фундаментом для подальшого використання методів штучного інтелекту, які розширюють можливості автоматичної оцінки та прогнозування стану людини. Поєднання цих підходів формує новий рівень розвитку цифрової медицини - від вимірювання до розуміння.

Застосування методів штучного інтелекту у системах моніторингу стану здоров'я

Штучний інтелект (ШІ) є одним із найважливіших напрямів розвитку сучасних технологій моніторингу здоров'я. Його впровадження дозволяє перейти від простого спостереження за фізіологічними показниками до їх інтелектуального аналізу, інтерпретації та прогнозування. На відміну від класичних статистичних методів, алгоритми ШІ здатні виявляти складні нелінійні закономірності у багатовимірних даних, адаптуватися до індивідуальних особливостей користувача і навчатися на основі досвіду. Це відкриває можливості для створення персоналізованих систем, які реагують не лише на абсолютні значення показників, а й на їхню динаміку у контексті звичної поведінки конкретної людини[19].

В основі застосування ШІ у сфері цифрової медицини лежить концепція «інтелектуального моніторингу» - автоматизованого процесу збирання, аналізу, інтерпретації та прогнозування даних про стан організму. У таких системах алгоритми виступають як аналітичний шар, що перетворює числові вимірювання на висновки про стан здоров'я, ризики або тренди. Це особливо актуально у випадках, коли дані надходять у великому обсязі з кількох сенсорів одночасно - наприклад, з фітнес-браслета, розумного термометра і смартфона.

Людина фізично не може оцінити такий потік інформації, а штучний інтелект здатний знаходити у ньому корисні закономірності.

Застосування ШІ у медичних системах умовно можна поділити на три основні напрями: виявлення аномалій, класифікація станів і прогнозування. Найбільш поширеною задачею у системах моніторингу є саме виявлення аномалій, коли алгоритм визначає, чи відповідають поточні дані «нормальній» поведінці організму. Для цього використовуються як алгоритми без учителя, так і гібридні моделі.

Одним із найбільш ефективних методів безнаглядного навчання є Isolation Forest (ліс ізоляцій). Цей алгоритм, запропонований у 2008 році, базується на ідеї випадкового поділу даних на множину дерев, у яких аномальні точки ізолюються швидше, ніж звичайні. У контексті моніторингу здоров'я Isolation Forest особливо зручний, оскільки не потребує попереднього маркування даних і може навчатися на реальних показниках користувача, які вважаються нормальними. Наприклад, якщо модель тренується на даних про серцевий ритм, сатурацію та температуру протягом кількох днів, вона створює власне уявлення про «індивідуальну норму». Коли з'являється нове вимірювання, алгоритм оцінює його за глибиною ізоляції у дереві - чим швидше спостереження відокремлюється, тим більш аномальним воно вважається. Перевагою цього підходу є висока швидкість, стійкість до шумів і можливість ефективно працювати навіть з невеликим набором даних, що робить його ідеальним для персоналізованих систем моніторингу.

Ще одним поширеним методом є Mahalanobis Distance - метрика, що враховує кореляції між параметрами та дозволяє виміряти «відстань» точки від центру розподілу даних. На відміну від звичайних евклідових відстаней, ця метрика зважає на співвідношення між змінними, тому здатна виявляти аномалії навіть тоді, коли окремі параметри виглядають нормальними. У практичному застосуванні до систем здоров'я це означає, що, наприклад, невелике підвищення пульсу у поєднанні з невеликим зниженням сатурації

може бути маркером втоми або початку гіпоксії - хоча жоден із параметрів окремо не виходить за межі норми. Таким чином, метод Mahalanobis дозволяє фіксувати комбіновані зміни, характерні для ранніх стадій фізіологічних порушень.

В останні роки значного поширення набули нейронні мережі - особливо рекурентні (RNN) та їхні модифікації, такі як LSTM (Long Short-Term Memory). Вони призначені для роботи з часовими рядами і здатні запам’ятовувати довготривалі залежності у послідовностях даних. Це надзвичайно корисно для моніторингу здоров’я, де показники мають виражений часовий характер. LSTM-мережі використовуються для прогнозування тенденцій - наприклад, передбачення підвищення температури або зміни серцевого ритму на кілька годин наперед. Їхня перевага полягає у здатності навчатися на історичних даних конкретного користувача, що відкриває шлях до повної персоналізації аналізу. Проте їхнє застосування обмежується високими вимогами до обчислювальних ресурсів і потребою у великому обсязі даних, тому такі алгоритми частіше використовуються у хмарних компонентах систем.

Іншим важливим напрямом розвитку є гібридні моделі, які поєднують класичні статистичні методи з елементами машинного навчання. Наприклад, система може спочатку застосовувати фільтрацію та статистичну нормалізацію для очищення сигналів, а потім передавати дані в алгоритм Isolation Forest або LSTM для аналізу. Такий підхід забезпечує баланс між точністю і швидкодією, дозволяючи отримувати результат у реальному часі без перевантаження обчислювальної системи. Гібридні моделі також можуть використовувати адаптивне оновлення - коли результати аналізу зворотно впливають на параметри фільтрації та пороги виявлення.

Значний внесок у підвищення точності оцінки роблять алгоритми кластеризації та самонавчання. Наприклад, алгоритм K-means може автоматично розділяти стан користувача на кілька груп - «норма», «легке відхилення», «аномалія». Після кожного нового дня роботи система оновлює

межі кластерів, поступово пристосовуючись до змін у способі життя користувача. Це дозволяє реалізувати ефект персоналізації без ручного налаштування. Такі системи стають «розумнішими» з часом, оскільки накопичують індивідуальні патерни поведінки організму.

Окреме місце займають моделі глибинного навчання на основі згорткових нейронних мереж (CNN), які ефективно працюють із сигналами, представленими у вигляді часово-частотних спектрів. Якщо PPG або ЕКГ-сигнал перетворити у спектрограму, CNN-мережа здатна автоматично розпізнавати характерні патерни - наприклад, зміни у формі хвилі, які передують аритмії. Такий підхід поступово наближає системи моніторингу до рівня автоматичної діагностики, де комп’ютер не лише повідомляє про відхилення, а й розпізнає їхню можливу причину.

У контексті персоналізованого моніторингу важливою перевагою ШІ є здатність адаптуватися. Алгоритми можуть навчатися на даних конкретного користувача, будуючи власну «модель норми». Це суттєво відрізняє такі системи від традиційних підходів, де всі користувачі оцінювалися за однаковими медичними порогами. Наприклад, у однієї людини пульс у 95 ударів може бути нормальною реакцією на легке навантаження, тоді як для іншої це вже ознака перевтоми. Завдяки ШІ система поступово «звикає» до індивідуальних параметрів і реагує лише на справжні відхилення.

Серед проблем, що залишаються актуальними у впровадженні ШІ у медичні системи, слід відзначити пояснюваність моделей. Для практичного застосування у медицині важливо не лише отримати результат «аномалія» або «норма», а й розуміти, чому алгоритм зробив такий висновок. Тому у сучасних дослідженнях активно розвиваються підходи Explainable AI (XAI), які дозволяють пояснювати внесок кожного параметра у прийняте рішення. Наприклад, система може вказати, що ризик був підвищений через надмірну варіабельність серцевого ритму або падіння SpO_2. Такий підхід підвищує довіру користувачів і медичних фахівців до результатів системи.

Ще одним напрямом є використання так званих «легких моделей» (AI-lite), призначених для пристроїв із обмеженими ресурсами. Ці алгоритми не потребують великих обчислювальних потужностей і працюють локально на самому пристрої, що дозволяє забезпечити автономність і конфіденційність. Вони можуть використовувати спрощені версії алгоритмів Mahalanobis або статистичні моделі, які відтворюють базові функції ШІ без складного навчання. Такі рішення є оптимальними для побутових і мобільних пристроїв.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ МОНІТОРИНГУ СТАНУ ЗДОРОВ'Я ЛЮДИНИ (SmartHealth)

2.1. Постановка задачі, призначення та функціональні вимоги до системи SmartHealth

Системи моніторингу стану здоров'я поступово перетворюються на один із ключових інструментів сучасної цифрової медицини. У міру зростання поширеності хронічних захворювань, серцево-судинних патологій, порушень дихання та постковідних ускладнень збільшується потреба у безперервному, персоналізованому контролі життєво важливих показників. Традиційні клінічні методи, що передбачають епізодичні вимірювання у медичних закладах, не дозволяють відстежувати динаміку змін у реальному часі та виявляти ранні ознаки фізіологічних відхилень. У цих умовах стає критично важливим створення інтелектуальних автоматизованих платформ, здатних збирати, аналізувати та інтерпретувати біофізіологічні сигнали постійно та з високою точністю.

Метою створення системи SmartHealth є розроблення інтелектуального апаратно-програмного комплексу, який забезпечує персоналізований моніторинг стану здоров'я користувача, виявлення відхилень на основі індивідуальних норм, а також їхню інтерпретацію у зручному й доступному для користувача форматі. На відміну від класичних систем, що покладаються на фіксовані порогові значення, SmartHealth повинен навчатися на даних конкретної людини, формуючи її унікальний фізіологічний профіль. Такий підхід дає змогу підвищити точність виявлення аномалій, мінімізувати кількість хибних спрацювань та адаптувати аналіз до індивідуальних особливостей організму.

Система призначена для широкої категорії користувачів: людей із хронічними захворюваннями, спортсменів, пацієнтів у період реабілітації, а також для звичайних користувачів, які прагнуть відслідковувати загальний рівень фізіологічного стану та попереджати можливі ризики. Окрім цього, система може бути розширена до роботи у медичних закладах, де дані пацієнтів

можуть інтегруватися у електронні медичні картки або використовуватися лікарями для дистанційного контролю.

Постановка задачі передбачає визначення повного циклу роботи системи SmartHealth. На першому етапі необхідно організувати збір фізіологічних даних у режимі реального часу. Оскільки в рамках даного дослідження застосовуються емулятори сенсорів, система повинна працювати із потоками пульсу, сатурації та температури, оновлюваними щосекунди. Майбутня версія системи може бути розширена до роботи з ЕКГ, акселерометрією, варіабельністю серцевого ритму, показниками дихання та іншими життєвими параметрами.

Другим етапом є проєктування надійного каналу передачі даних і створення внутрішньої бази даних, у якій усі вимірювання будуть зберігатися у вигляді часових рядів. База даних має забезпечувати швидкий доступ до великих обсягів даних за останні хвилини, години або добу, що дозволить системі виконувати обчислення в реальному часі, будувати статистичні моделі та оновлювати персоналізовані оцінки норм.

Після збирання даних перед системою стоїть задача попередньої обробки: фільтрація шумів, нормалізація, виявлення пропусків та перевірка фізіологічної достовірності значень. Оскільки сигнали, такі як пульс або сатурація, можуть містити артефакти, спричинені рухом, освітленістю або іншими зовнішніми факторами, важливо щоб перед застосуванням алгоритмів аналізу система виконувала адаптивну стабілізацію сигналу. Це створює основу для подальшого інтелектуального аналізу.

На центральному етапі функціонування SmartHealth відбувається аналітична оцінка стану користувача. Тут система застосовує два типи алгоритмів: класичні статистичні методи (ковзні медіани, коефіцієнти відхилення, Z-score, MAD) та алгоритми штучного інтелекту, такі як Isolation Forest і Mahalanobis Distance. Використання обох підходів дозволяє побудувати гібридну модель аналізу: статистичні методи забезпечують швидку та просту

оцінку сигналів у реальному часі, тоді як AI-модуль додає здатність до глибокої персоналізованої інтерпретації. Важливо, щоб система могла працювати у двох режимах: з повноцінною моделлю Isolation Forest (якщо доступна відповідна інфраструктура) та у легкому режимі AI-lite (якщо ресурси обмежені або середовище розгортання не підтримує складні моделі).

Одним з ключових компонентів SmartHealth є модуль персоналізації. Він формує для кожного користувача індивідуальну базову лінію по всіх параметрах, враховуючи добові ритми, адаптацію до фізичного навантаження, типову поведінку та характер трендів. Це дозволяє системі реагувати не на абсолютні значення, а на відхилення від звичної для конкретної людини динаміки. Такий підхід значно підвищує якість аналізу у реальних умовах, де універсальні пороги недостатньо інформативні.

Останнім етапом є візуалізація та надання інформації користувачу. Система повинна відображати дані у зрозумілому вигляді, надаючи інструменти для перегляду графіків, аномалій, пояснення рішень та зведених показників. Дашборд має бути інтуїтивно зрозумілим, адаптивним та оновлюватись у реальному часі. Особлива увага приділяється пояснюваності результатів - користувач повинен бачити, чому система визначила певний стан як аномальний, які параметри внесли найбільший вклад у рішення та наскільки високою є невизначеність оцінки.

У рамках формулювання вимог до системи виділяють функціональні та нефункціональні вимоги. До функціональних належить можливість безперервного збору даних, їх збереження у базі, статистичного та інтелектуального аналізу, автоматичного виявлення відхилень, зручного інтерфейсу для перегляду історії та пояснень. Нефункціональні вимоги включають стабільність роботи, стійкість до шумів, низьке енергоспоживання у перспективі апаратних реалізацій, захищеність даних, масштабованість та можливість розширення. У майбутньому система повинна підтримувати

інтеграцію зі стандартами медичної інтероперабельності, такими як HL7 або FHIR, що дозволить використовувати SmartHealth у клінічних середовищах.

Таким чином, система SmartHealth повинна відповідати вимогам персоналізованої цифрової медицини, об’єднуючи апаратне та програмне забезпечення у єдину інтелектуальну платформу. Вона має забезпечити надійність, точність, адаптивність та зручність використання, а також містити механізми для подальшого розвитку і розширення функціональності.

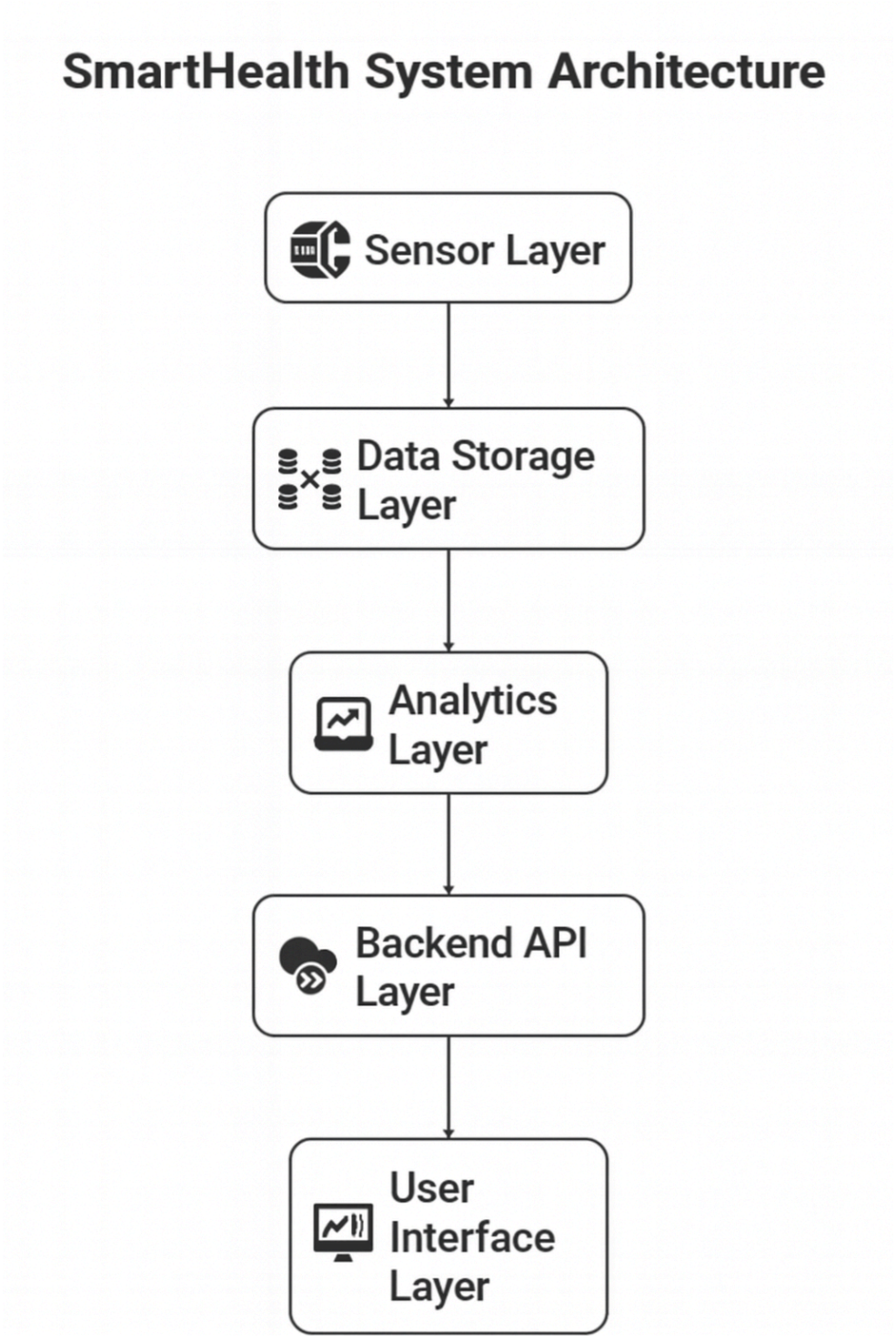
2.2. Загальна архітектура та концепція побудови системи SmartHealth

Архітектура системи SmartHealth базується на принципах модульності, масштабованості та інтегрованості, що дозволяє забезпечити надійну роботу в умовах безперервного збору даних та їх реального аналізу. Система розроблена як багатошарова платформа, у якій кожен шар відповідає за окрему групу функцій - від отримання фізіологічних сигналів до застосування алгоритмів штучного інтелекту та візуалізації результатів у зручному інтерфейсі.

Концепція побудови SmartHealth передбачає виділення чотирьох основних рівнів: сенсорного, транспортного, аналітичного та користувацького. Такий підхід дозволяє розділити відповідальності між компонентами та забезпечити незалежність їхнього розвитку. У разі зміни технологічної бази (наприклад, підключення нових сенсорів або заміна алгоритму ШІ), інші компоненти можуть залишатися незмінними, що істотно підвищує гнучкість системи.

На найнижчому рівні розміщується сенсорний шар, відповідальний за збір фізіологічних даних. У рамках реалізованої версії SmartHealth дані генерує програмний емулятор сенсорів, який імітує показники частоти серцевих скорочень, сатурації крові та температури. У повній апаратній реалізації система може працювати з реальними PPG-датчиками, інфрачервоними термометрами, акселерометрами або іншими біомедичними сенсорами. Сенсорний шар передає дані з фіксованою частотою в один або кілька потоків, де кожне спостереження має часову мітку(Рисунок 1).

Рисунок 1 - Загальна архітектура SmartHealth



Над сенсорним рівнем розташовано шар передачі та акумуляції даних. Він відповідає за приймання потоків вимірювань, їх структурування та збереження у локальній базі даних. У реалізації SmartHealth використано SQLite як легковагову систему керування базами даних, що забезпечує високу швидкість операцій читання-запису та не потребує окремого сервера. База містить

таблицю vitals(Рисунок 2), до якої записуються усі вимірювання з прив’язкою до користувача та часу. Така структура дозволяє ефективно виконувати операції вибірки за останні хвилини або годинами, що необхідно для динамічного аналізу.

Рисунок 2 - *Логічна схема бази даних SmartHealth*

Vitals Table

Field	Type	Description
id	INTEGER	Primary key, auto-incrementing integer
ts	REAL	UNIX timestamp of the measurement
hr	REAL	Heart rate (beats per minute)
spo2	REAL	Blood oxygen saturation (%)
temp	REAL	Body temperature (°C)
user_id	TEXT	Identifier of the user

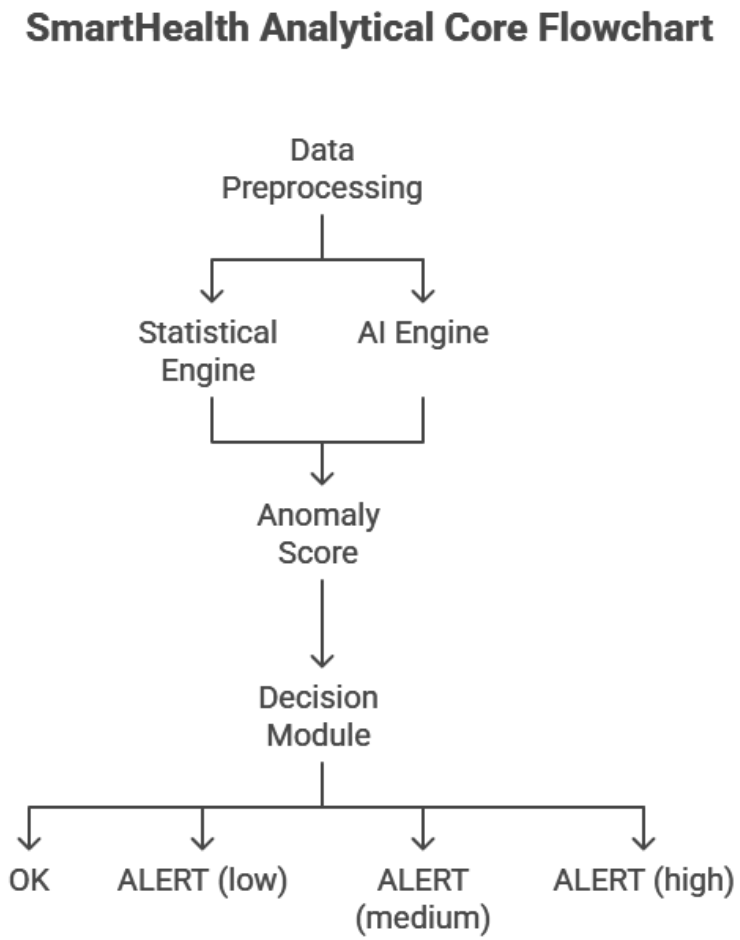
Наступний рівень - шар попередньої обробки та нормалізації. На цьому етапі дані перевіряються на наявність шумів, артефактів і пропусків. Оскільки біосигнали часто мають природну варіабельність і можуть бути спотворені зовнішніми факторами, система застосовує згладжування, ковзні медіани та базові статистичні фільтри. Це необхідно для забезпечення стабільної роботи аналітичного ядра, оскільки нечіткі або нестабільні дані різко знижують якість виявлення аномалій.

Центральним елементом архітектури є аналітичний модуль, що поєднує класичні статистичні методи і алгоритми штучного інтелекту. Його робота базується на формуванні персональної базової лінії користувача, обчисленні індивідуальних Z-score, виявленні відхилень та застосуванні моделей для оцінки «анормальності» параметрів. У системі SmartHealth побудовано два

режими роботи: повноцінний режим із застосуванням Isolation Forest та легкий режим AI-lite, що використовує Mahalanobis Distance і статистичні ознаки. Це дозволяє адаптувати систему до умов середовища, у якому вона працює: якщо Python-платформа підтримує scikit-learn, використовується більш точний метод, якщо ні - система переходить у легкий режим.

Характерною особливістю аналітичного ядра є його персоналізованість. На відміну від статичних порогових моделей, які однаково оцінюють усіх користувачів, SmartHealth будує модель для кожної людини окремо(Рисунок 3). Це досягається шляхом накопичення даних за певний проміжок часу (наприклад, 10 хвилин) та аналізу гісторичних характеристик. Таким чином система «звикає» до звичних фізіологічних значень конкретного користувача, а відхилення визначає відносно цих даних, а не за універсальними медичними нормами.

Рисунок 3 - Архітектура аналітичного ядра



Важливою частиною архітектури є API-шар, що реалізує взаємодію між клієнтськими інтерфейсами та сервером. SmartHealth використовує FastAPI як високопродуктивний веб-фреймворк, який забезпечує швидку передачу даних, низькі затримки та можливість одночасного опрацювання багатьох запитів. Через API реалізовано отримання останніх вимірювань, вибір часового діапазону, тренування моделі, перевірку статусу моделі, отримання пояснень та графічних даних. Така архітектурна побудова дозволяє легко масштабувати систему та додавати нові компоненти без втручання в основу.

Останній рівень - користувацький інтерфейс. У цій реалізації SmartHealth застосовано вбудований веб-дашборд на основі JavaScript та Chart.js, а також можливість розширення через Streamlit. Дашборд відображає ключові фізіологічні параметри користувача, їх часові графіки, індикатори стану, ступінь відхилення, пояснення рішень та прогнозні оцінки. Важливим елементом є візуальні індикатори аномалій, які дозволяють одразу оцінити стан користувача. Графіки оновлюються в реальному часі, а дані подаються в адаптивному форматі.

Загалом архітектура системи побудована за принципом «від сенсора до інтелектуальної інтерпретації». Така концепція підкреслює, що основна цінність системи полягає не у зборі даних, а саме у їх глибокому аналізі та поясненні. Кожен компонент архітектури SmartHealth розроблено таким чином, щоб забезпечити максимальну надійність та адаптивність, дозволяючи системі працювати як у локальному середовищі, так і у хмарному, та бути готовою до підключення будь-яких фізіологічних датчиків у майбутньому.

Архітектурний підхід, закладений у SmartHealth, дозволяє розглядати систему як платформу, здатну до подальшого розширення: підключення нових сенсорів, впровадження розширених AI-модулів, інтеграція з мобільними додатками або медичними інформаційними системами. Завдяки модульній структурі система може гнучко адаптуватися під різні сценарії використання - від домашнього самоконтролю до клінічних застосувань.

2.3. Вибір моделі розробки та організація процесу створення системи SmartHealth

Проєктування інтелектуальної системи моніторингу стану здоров’я SmartHealth потребує вибору такої моделі життєвого циклу програмного забезпечення, яка забезпечить гнучкість, стабільність та можливість поетапного нарощування функціональності. Оскільки система включає кілька взаємопов’язаних компонентів - шар отримання фізіологічних сигналів, базу даних, серверну частину, аналітичний модуль та веб-інтерфейс користувача - модель розробки має дозволяти незалежний розвиток цих частин, а також зручну їх інтеграцію. Серед класичних і сучасних моделей розробки програмних систем найбільш поширеними є каскадна, V-модель, модель прототипування, спіральна модель, гнучкі методології (Agile/Scrum) та ітеративно-інкрементний підхід. Їх порівняння подано у таблиці 2.

Таблиця 2 - Порівняльний аналіз моделей розробки програмного забезпечення

Модель розробки	**Переваги**	**Недоліки**	**Доцільність використання**
Каскадна (Waterfall)	Чітка структура; проста документальність	Низька гнучкість; складність внесення змін	Вузькі проєкти з фіксованими вимогам
V-модель	Посилений фокус на тестуванні; відповідність етапів	Та сама негнучкість, що й водоспад	Критичні системи з фіксованими вимогами
Модель прототипування	Швидке створення демонстрацій; зручний UX-підхід	Прототип може відрізнятися від фінальної системи	Системи з високою невизначеністю вимог
Спіральна модель	Орієнтація на ризики; гнучкість	Складність застосування; висока вартість	Великі комплексні проєкти

Agile / Scrum	Висока адаптивність; швидкі релізи	Потребує команди; не завжди підходить для НДР	Комерційні проєкти, стартапи
Ітеративно-інкрементна модель	Гнучкість; поетапне нарощування функцій; стабільність архітектури; зручність для ML	Потребує продуманої архітектури на старті	Наукові дослідження, ML/AI-проєкти

Для системи SmartHealth найбільш придатною є ітеративно-інкрементна модель, оскільки вона комбінує системність і гнучкість, дозволяє реалізовувати складні аналітичні модулі, проводити тестування після кожної ітерації та поетапно інтегрувати нові функції. Такий підхід забезпечує можливість швидкого прототипування статистичних методів, поступового впровадження алгоритмів штучного інтелекту, вдосконалення архітектури API, адаптації структури бази даних та оптимізації інтерфейсу користувача.

Розвиток SmartHealth фактично відбувався у кілька послідовних ітерацій. Перша ітерація була зосереджена на побудові мінімально працездатного прототипу (MVP), який включав структуру бази даних, серверну частину FastAPI з базовими кінцевими точками (/ingest, /latest), початкову реалізацію фіксації даних та найпростіший інтерфейс. Друга ітерація була спрямована на доповнення статистичними методами аналізу - методом Z-score, Median Absolute Deviation, розрахунком ковзних медіан та формуванням базових правил виявлення критичних станів. На третьому етапі система була доповнена модулем машинного навчання на основі алгоритму Isolation Forest, а також альтернативним обчислювально легким механізмом AI-lite (Mahalanobis Distance), що дозволяє працювати за відсутності бібліотеки scikit-learn. Четверта ітерація включала повне оновлення користувацького інтерфейсу, додавання візуалізації показників у реальному часі, блоку пояснюваності моделі та загального рефакторингу фронтенду. На п’ятій ітерації виконано інтеграцію

всіх компонентів, оптимізацію продуктивності, логування та підготовку системи до розгортання.

Ітеративно-інкрементна модель дозволила організувати життєвий цикл SmartHealth як послідовність фаз: аналіз вимог → проєктування → реалізація → тестування → розгортання → удосконалення. На кожному етапі система залишалася працездатною, що дозволяло контролювати стабільність і якість рішень. Оскільки SmartHealth має модульну архітектуру, кожен цикл працював із конкретним підкомпонентом системи - базою даних, API, аналітичним ядром або інтерфейсом. Це забезпечило можливість локальних оновлень без руйнування цілісності продукту.

Організація процесу розробки в межах обраної моделі включала такі ключові види діяльності: визначення цілей і функцій для кожної ітерації; проєктування архітектурних рішень; реалізація відповідних компонентів; тестування модулів; інтеграція та аналіз результатів; рефакторинг та підготовка до наступної ітерації. Особливу роль відіграв постійний аналіз роботи алгоритмів машинного навчання, оскільки ці модулі потребують емпіричного коригування на основі тестових і реальних даних.

Таким чином, ітеративно-інкрементна модель забезпечила SmartHealth оптимальну гнучкість та науково-технічну обґрунтованість, дозволивши розвивати систему у логічних етапах, підтримувати її працездатність на кожному етапі і поступово інтегрувати нові елементи функціональності.

2.4. Обґрунтування вибору інструментальних засобів розробки системи SmartHealth

Розробка інтелектуальної системи моніторингу стану здоров’я SmartHealth потребувала застосування сучасних, надійних та ефективних інструментальних засобів, здатних забезпечити збір, обробку, зберігання, аналіз фізіологічних даних і відображення результатів у реальному часі. Особливістю проєкту є поєднання кількох важливих компонентів - бекенд-серверної частини, аналітичного ядра, модуля машинного навчання, локальної бази даних, а також

інтерактивного інтерфейсу користувача. У зв’язку з цим доцільно здійснити техніко-економічне обґрунтування вибору інструментів, які забезпечили б високу продуктивність, простоту розробки, масштабованість та надійність.

При виборі інструментів було враховано такі критерії: відкритість і доступність технологій, кросплатформеність, продуктивність в обробці потокових даних, наявність бібліотек для статистичного аналізу та машинного навчання, зручність інтеграції між компонентами, підтримка швидкої розробки REST API, простота розгортання та невибагливість до обчислювальних ресурсів. У цьому підрозділі подано детальний огляд інструментів, їх порівняння та обґрунтування вибору для реалізації SmartHealth.

Вибір мови програмування: Python

Ключовим інструментом розробки серверної частини SmartHealth стала мова Python, що пояснюється її широким використанням у сфері роботи з даними, машинного навчання та медичних інформаційних систем. Python забезпечує надзвичайно розвинену екосистему бібліотек для статистики, аналітики, моделювання, побудови API та взаємодії з базами даних.

Переваги Python можна сформулювати так:

- наявність зрілих наукових бібліотек (NumPy, SciPy, pandas);
- розвинена підтримка машинного навчання (scikit-learn, TensorFlow, PyTorch);
- зручність створення веб-сервісів завдяки FastAPI та Flask;
- велика кількість інструментів для обробки тимчасових рядів;
- уніфікованість коду та швидка розробка прототипів;
- велика спільнота та наявність рішень для практично будь-яких задач.

Окрім того, Python дозволяє гнучко масштабувати систему: від локальних серверів до хмарних контейнеризованих інсталяцій.

Вибір фреймворку для серверної частини: FastAPI

Для реалізації API системи SmartHealth обрано фреймворк FastAPI, який є сучасною, високопродуктивною платформою для створення RESTful-сервісів та потокової обробки даних. FastAPI дозволяє легко реалізувати високонавантажені кінцеві точки, обробку JSON-структур, автоматичну валідацію входу та виходу, а також швидке масштабування.

Основні переваги FastAPI:

- висока продуктивність завдяки використанню ASGI-сервера Uvicorn;
- автоматична генерація документації OpenAPI/Swagger;
- підтримка асинхронних запитів - важливо для роботи з потоками сенсорних даних;
- інтеграція з Pydantic, що забезпечує типізацію та валідацію даних;
- зручність створення модульної структури коду;
- низькі затрати ресурсів - система ефективно працює навіть на малопотужних ПК.

FastAPI дозволив створити ендпоінти /ingest, /latest, /series, /train, /train_status з мінімальними витратами часу та забезпечити швидку реакцію інтерфейсу SmartHealth у реальному часі.

Обґрунтування вибору бази даних: SQLite

Для локального зберігання показників життєвих параметрів використано базу даних SQLite. Це легка, вбудована реляційна СУБД, яка не потребує

серверного середовища, легко інтегрується з Python і забезпечує високу швидкість та надійність.

Переваги SQLite:

- відсутність необхідності встановлювати окремий сервер БД;
- низькі накладні витрати;
- оптимальна робота з невеликими та середніми обсягами даних;
- атомарність транзакцій та надійність збереження;
- простота резервного копіювання - база представлена одним файлом;
- зручність для тестування моделей ML та прототипування.

У контексті SmartHealth SQLite забезпечує ефективну роботу з часовими рядами, дозволяючи у режимі реального часу вибирати останні 30 хвилин даних, а також зберігати історію показників для подальшого аналізу.

Вибір бібліотек для аналітики та статистичної обробки

Система SmartHealth потребує аналізу фізіологічних сигналів у режимі реального часу, обчислення ковзних статистик, оцінювання відхилень та виявлення аномалій. Для цього застосовано такі бібліотеки:

NumPy - основна бібліотека для роботи з масивами та векторними обчисленнями. Використовується для розрахунку основних статистик, нормування, фільтрації та підготовки даних.

pandas - забезпечує високу продуктивність при роботі з часовими рядами, дозволяє виконувати групування даних, агрегування та обчислення ковзних значень (rolling window).

SciPy - використовується для статистичних методів та моделювання, хоча у системі SmartHealth основні завдання реалізовано засобами NumPy та pandas.

Усі бібліотеки працюють узгоджено і забезпечують швидку обробку навіть потокових даних.

Вибір інструментів для машинного навчання

Система SmartHealth застосовує два підходи:

1. повноцінний алгоритм Isolation Forest (якщо доступний scikit-learn)
2. обчислювально легкий режим AI-lite (Mahalanobis Distance)

Для повної підтримки алгоритмів машинного навчання використовується бібліотека scikit-learn, яка включає:

- алгоритми для класифікації, кластеризації та виявлення аномалій;
- зручний API для навчання та застосування моделей;
- підтримку збереження моделей у форматі joblib;
- механізми масштабування та попередньої обробки даних.

Алгоритм Isolation Forest є оптимальним для персоналізованих систем моніторингу здоров’я, оскільки не потребує маркованих даних і здатний працювати навіть із невеликими наборами спостережень.

Як резервний варіант застосовується власноруч реалізований метод Mahalanobis Distance, який не залежить від зовнішніх бібліотек ML і дозволяє забезпечити роботу аналітичного блоку у випадку обмежень середовища або недоступності scikit-learn.

Вибір засобів для реалізації веб-інтерфейсу

Користувацький інтерфейс SmartHealth представлений інтерактивним дашбордом, який відображає графіки фізіологічних сигналів, показники аномалій, пояснення внеску ознак та статус роботи моделі. Дашборд створено із застосуванням таких технологій:

HTML + CSS - для структури та стилізації інтерфейсу. Використано сучасні підходи до адаптивного дизайну, включаючи CSS Grid та Flexbox.

JavaScript - забезпечує інтерактивну логіку, оновлення графіків у реальному часі та періодичні запити до API.

Chart.js - бібліотека для відображення графіків. Підтримує лінійні графіки з високою частотою оновлення, що критично важливо для візуалізації фізіологічних сигналів.

Fetch API - використовується для взаємодії з бекендом у режимі реального часу з інтервалом у 2 секунди.

Таке рішення дозволило створити легкий, стабільний та інтуїтивно зрозумілий інтерфейс.

Інструментальні засоби для розгортання та тестування

Для розробки та тестування використовувався редактор Visual Studio Code, який забезпечив:

- зручна структура проєкту;
- вбудований інтегрований термінал;
- автоматичне форматування коду;
- підсвітка синтаксису Python, HTML, JS;
- широкий набір розширень для FastAPI та Python.

Тестування виконувалося як вручну (інтерактивне спостереження за дашбордом), так і програмно - шляхом моделювання потоків даних в API /ingest.

Аргументація вибору загальної комбінації інструментів

Усі обрані засоби утворюють узгоджену технологічну екосистему, що забезпечує:

- високу швидкість розробки;
- простоту інтеграції між компонентами;
- надійність зберігання даних;
- ефективність алгоритмів аналізу;
- можливість персоналізації моделей;
- легкість розгортання у локальному або хмарному середовищі.

Комбінація Python + FastAPI + SQLite + scikit-learn + Chart.js є оптимальною для реалізації інтелектуальної системи моніторингу стану здоров’я, яка повинна працювати у реальному часі та забезпечувати персоналізовану оцінку стану користувача.

2.5. Основні режими функціонування та програмна реалізація системи SmartHealth

Система SmartHealth функціонує як комплексний, багатокомпонентний програмний інструмент для безперервного відстеження, аналізу та інтерпретації фізіологічних показників користувача у реальному часі. У межах даного розділу подано розгорнутий опис принципів роботи системи, архітектури її компонентів, особливостей обробки даних, а також механізмів візуалізації, тренування моделей машинного навчання та взаємодії між сервером, базою даних і користувацьким інтерфейсом.

Основна ідея SmartHealth полягає у створенні адаптивної, персоналізованої системи, здатної поступово формувати уявлення про індивідуальні параметри фізіологічної “норми”, що є вирішальним фактором у ранньому виявленні потенційних ризиків для здоров’я. Реалізація системи відбувається через поєднання серверної частини на базі FastAPI, локальної бази даних SQLite, аналітичного ядра, реалізованого методами статистики та

алгоритмами машинного навчання, а також вебдодатка, що формує інтерактивний дашборд користувача.

Програмна система функціонує у декількох важливих режимах: режимі збору даних, режимі аналізу, режимі зберігання, режимі візуалізації, режимі автоматичного визначення аномалій та режимі тренування персоналізованої моделі. Кожен із цих режимів реалізований через спеціалізовані модулі, які разом формують замкнутий цикл обробки інформації.

У структурі системи ключову роль відіграє серверна частина. Сервер відповідає за прийом даних від сенсорів або емулятора, здійснює їх первинну перевірку, зберігає їх у базу даних, а також запускає обчислювальні механізми, які генерують оцінку поточного стану користувача. У SmartHealth сервер працює у форматі REST API, що дозволяє легко масштабувати систему, додавати нові сенсори, розширювати перелік показників, інтегрувати зовнішні сервіси. Комунікація між компонентами відбувається виключно через HTTP-запити, що забезпечує модульність і легкість розгортання.

Усі фізіологічні дані зберігаються у локальній базі даних SQLite. Це оптимальне рішення для персональних систем моніторингу здоров’я, оскільки SQLite не потребує окремого сервера, працює надзвичайно швидко для часових рядів та не ускладнює інсталяцію програмного забезпечення. Таблиця зберігає часові мітки, значення пульсу, сатурації, температури та ідентифікатор користувача. Таке рішення забезпечує можливість одночасної роботи з кількома користувачами та реалізацію персоналізованих моделей.

Аналітичне ядро SmartHealth - це найважливіша складова системи, оскільки саме вона дозволяє здійснювати глибинний аналіз фізіологічних сигналів. Ядро складається з двох паралельних підсистем: статистичної та підсистеми машинного навчання. Статистична частина виконує оперативний аналіз показників через методи ковзного вікна, медіанних згладжувань, обчислень MAD (Median Absolute Deviation), стандартних Z-score та порівняння зі встановленими фізіологічними порогами. Вона працює миттєво і не потребує

тренування моделей, тому використовується як механізм базового контролю та первинної діагностики.

Паралельно з цим працює алгоритмічна підсистема, орієнтована на виявлення індивідуальних закономірностей. У SmartHealth застосовано два режими машинного навчання. Перший - Isolation Forest. Цей алгоритм є ідеальним для задач виявлення аномалій, оскільки він не потребує маркованих даних і автоматично розрізняє типові та нетипові патерни поведінки фізіологічних показників. Другий режим - AI-lite - базується на відстані Махаланобіса, яка дозволяє оцінювати ступінь відхилення поточного стану від статистично сформованої «норми» навіть тоді, коли повноцінне машинне навчання недоступне. Завдяки використанню AI-lite система може працювати в умовах обмежених обчислювальних можливостей.

Особливістю SmartHealth є механізм тренування персоналізованої моделі. Користувач може вручну натиснути кнопку «Перетренувати модель», після чого система виконує повний цикл побудови ознак (feature extraction), формує матрицю ознак на ковзних вікнах довжиною 60 секунд, тренує Isolation Forest та зберігає модель разом із метаданими - часом тренування, набором параметрів, кількістю вікон. Це формує індивідуальну модель кожного користувача, яка враховує його власну фізіологічну динаміку, рівень активності, коливання пульсу та температури. Такий підхід максимально відповідає сучасним тенденціям Precision Health - медицини точного здоров’я.

Користувацький інтерфейс SmartHealth реалізовано як вебдодаток, що працює у браузері та використовує сучасні засоби побудови інтерактивних графіків. Інтерфейс побудовано на основі Chart.js, що дозволяє легко масштабувати графіки, відображати великі масиви даних, накладати порогові лінії (наприклад, HR=140 або SpO_2=90%), виконувати інтелектуальний даунсемплінг часових рядів та підтримувати динамічну зміну користувача. Інтерфейс містить панель стану, блок із поясненням рішень ШІ (explainability),

графіки основних показників та окремі блоки з інформацією про score, z-score та рівень невизначеності.

Інтерфейс веб-дашборду SmartHealth реалізовано у вигляді інтерактивної односторінкової панелі моніторингу, яка забезпечує відображення поточних фізіологічних показників користувача, аналітичних висновків моделі машинного навчання, прогнозу стану та графіків часових рядів у режимі реального часу. Оновлення даних здійснюється автоматично кожні дві секунди за допомогою запитів до бекенд-сервера на базі FastAPI.

1. Панель вибору користувача

У верхній частині дашборду розміщено елемент вибору *user_id*, який відображає перелік доступних користувачів, наявних у базі даних. Поруч розташована кнопка оновлення списку користувачів, а також перемикач запуску повторного тренування моделі (доступний лише за наявності модуля scikit-learn). Тут же знаходиться індикатор загального стану користувача (*OK* або *ALERT*), який формується на основі результатів роботи алгоритмів аналізу даних.

2. Блок поточного стану

Блок містить такі показники:

- Severity - рівень тривожності стану (*low*, *medium*, *high*);
- AI score - числовий індикатор аномальності, сформований моделлю Isolation Forest або алгоритмом Mahalanobis у режимі AI-lite;
- Uncertainty - рівень невизначеності рішення моделі.

Ці дані отримуються через API-ендпоінт /latest. Значення AI score, близькі до нуля, свідчать про відсутність аномалій; чим більш від’ємним є показник, тим підозрілішим вважається стан. Показник невизначеності визначається величиною AI score та наявністю тренованої персоналізованої моделі.

3. Блок Z-scores

Z-оцінки використовуються для визначення персональних відхилень фізіологічних сигналів від індивідуальної норми користувача. Розрахунок здійснюється на основі ковзних медіанних значень та robust-дисперсії MAD. Використовується формула:

$$z = \frac{x - median}{1.4826 \cdot MAD}$$

Це забезпечує стійкість до шумів і дозволяє системі адаптуватися до типових фізіологічних діапазонів конкретної людини, а не до загальних норм. У текстовому поясненні зазначається, наскільки кожен показник (пульс, сатурація, температура) відхиляється від індивідуального діапазону.

4. Блок Explain (внесок ознак)

Блок пояснюваності відображає, який показник найбільше вплинув на поточну оцінку стану. Після обчислення абсолютних значень Z-оцінок система нормалізує їх і формує відсотковий внесок кожної ознаки:

$$w_k = \frac{|z_k|}{\sum |z|}$$

Таким чином користувач отримує зрозумілу інтерпретацію: який саме фізіологічний параметр став ключовим фактором оцінки моделі. Це підвищує прозорість роботи алгоритму та відповідає сучасним вимогам до Explainable AI.

5. Аналітичний блок

Цей блок узагальнює результати роботи системи та містить:

- визначений моделью стан (*OK* або *ALERT*);
- пояснення рівня невизначеності рішення;
- текстове інтерпретування AI score;
- домінуючу ознаку, що вплинула на рішення.

Аналітичний блок виступає основним джерелом інтерпретації результатів для кінцевого користувача, оскільки комплексно описує стан організму на основі сукупності сигналів.

6. Блок прогнозування стану

Для короткострокового прогнозу використано експоненціальне згладжування (Exponential Smoothing), яке дозволяє оцінити очікувані значення фізіологічних показників у найближчі хвилини:

$$F_{t+1} = \alpha x_t + (1 - \alpha) F_t$$

де $\alpha = 0.5$.

Прогноз формується окремо для пульсу, сатурації та температури й відображається текстово.

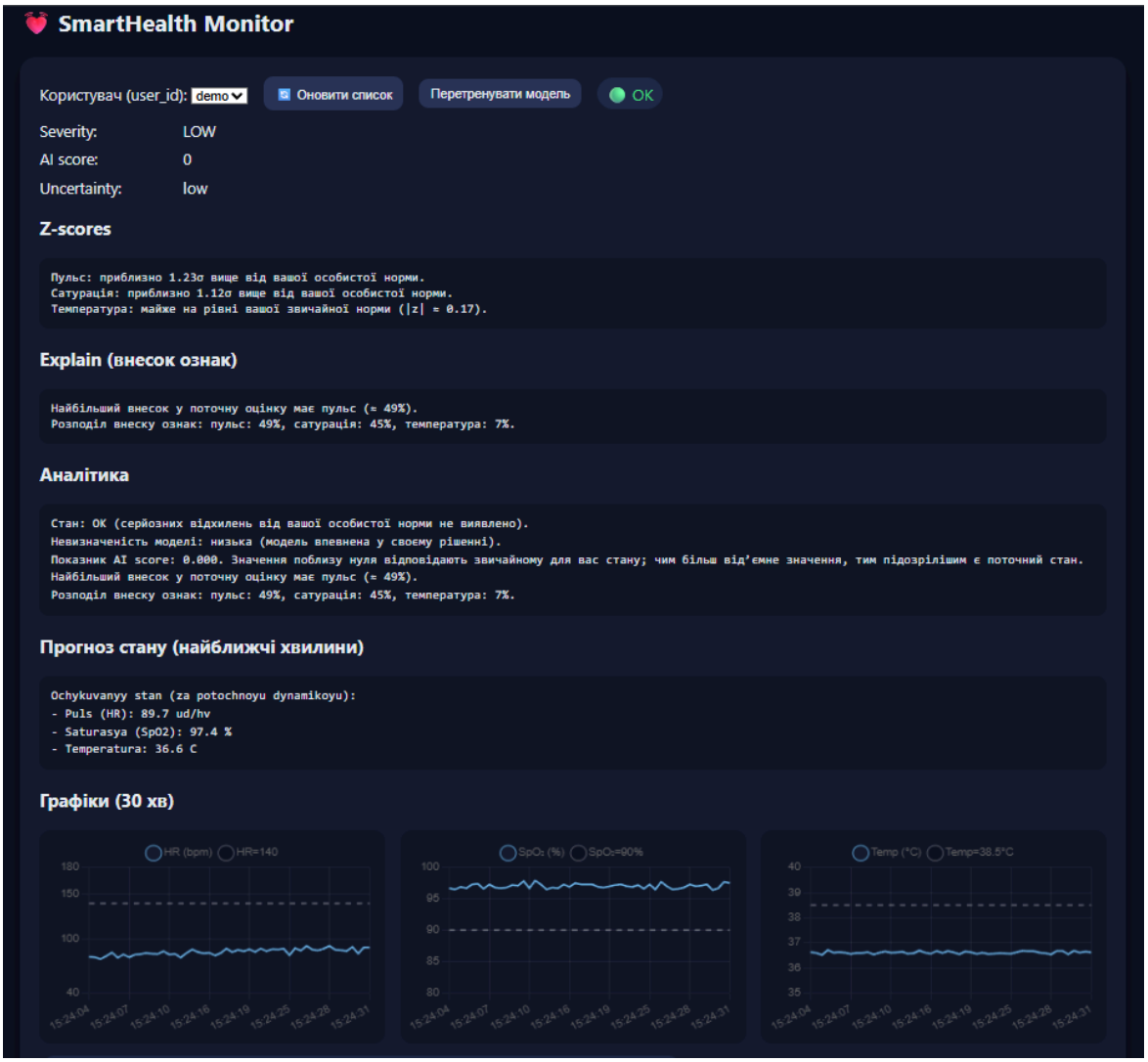
7. Графіки часових рядів

Графічний модуль побудовано на базі бібліотеки Chart.js. Дашборд відображає три часові ряди за останні 30 хвилин:

- частота серцевих скорочень (HR),
- рівень кисню в крові (SpO_2),
- температура тіла.

Для кожного сигналу на графіку нанесено порогові значення (наприклад, HR=140, SpO_2=90 %, температура=38.5 °C), що дає можливість візуально оцінити, чи перетинають дані критичні межі. Графіки оновлюються в реальному часі на основі даних ендпоінту /series.

Рисунок 4 - Інтерфейс SmartHealth



Дані оновлюються автоматично кожні 2 секунди, що дозволяє відстежувати зміни у реальному часі. Особлива увага приділяється індикації стану: зелений індикатор відображає норму, тоді як червоний сигналізує про тривогу. Це дозволяє користувачу швидко оцінити власний стан. У разі аномалії система вказує на її джерело: наприклад, домінуючий внесок пульсу або сатурації(Рисунок 4).

Усі режими роботи SmartHealth взаємопов'язані та працюють безперервно. Режим збору даних забезпечує регулярне поповнення бази. Режим аналізу автоматично виконує оцінку кожної нової точки. Режим візуалізації гарантує відображення останнього стану. Режим тренування дозволяє покращити точність системи відповідно до індивідуальних характеристик

користувача. У випадку відсутності scikit-learn автоматично активується режим AI-lite, що зберігає функціональність системи.

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ СИСТЕМИ SMARTHEALTH

3.1 Мета та постановка експерименту

Метою експериментального дослідження є комплексна перевірка працездатності системи SmartHealth, оцінка точності реалізованих аналітичних алгоритмів та визначення їхньої надійності в умовах реального часу. Експеримент спрямований на підтвердження коректності обробки фізіологічних сигналів (пульсу, сатурації та температури), адекватності реакції на нормальні та аномальні стани, а також здатності системи до персоналізації та адаптивного прогнозування. Для цього було створено спеціальний програмний емулятор, який генерує послідовності фізіологічних показників у різних режимах: нормальному, прикордонному та патологічному. Дані надходять у систему з частотою 1 Гц, записуються до локальної бази SQLite та обробляються бекендом у режимі реального часу.

У межах експерименту перевіряється декілька ключових аспектів: стійкість обчислення ковзних статистик (медіани, MAD), правильність формування Z-score, робота алгоритмів Mahalanobis distance та AI-lite, поведінка моделі Isolation Forest після персоналізованого тренування, а також коректність обчислення прогнозу фізіологічних показників за допомогою експоненційного згладжування. Додатково тестується швидкодія інтерфейсу, стабільність відображення графіків, відповідність поясненості рішень очікуваним результатам та реакція системи на затримки чи нерегулярність потоку даних.

Таким чином, постановка експерименту охоплює перевірку всієї архітектури SmartHealth: від збору та накопичення показників до виявлення аномалій, інтерпретації стану користувача, формування прогнозу та відображення результатів у дашборді. Результати експериментального дослідження дозволяють оцінити точність, надійність та практичну придатність

розробленої системи, а також визначити переваги й обмеження застосованих алгоритмічних рішень.

3.2 Опис програмної реалізації експерименту

Основою експерименту є процедура надсилання симульованих вимірювань із частотою один раз на секунду, що відповідає типічним параметрам роботи побутових та клінічних пристроїв моніторингу. Дані потрапляють у бекенд через REST-метод */ingest*, після чого зберігаються у таблиці vitals(Рисунок 2) бази SQLite. У процесі накопичення формується 30-хвилинне ковзне вікно, з якого система отримує останні значення показників, обчислює ковзні статистики (медіану, MAD), визначає Z-score та аналізує тренди. Паралельно система здійснює побудову ознак для 60-секундного вікна, необхідних для роботи алгоритмів Mahalanobis distance та моделі Isolation Forest.

У межах експерименту особлива увага приділялась реалізації механізмів персоналізації. Для користувача, який накопичив більше десяти хвилин історії, активується можливість навчання моделі Isolation Forest. Сам процес тренування відбувається у кілька етапів: побудова фічевої матриці на основі ковзних вікон, навчання моделі, збереження ваг у файловій системі та фіксація метаданих (кількість вікон, часовий діапазон, набір ознак). Після завершення тренування система автоматично переходить до режиму інференсу, де для кожної нової точки даних викликається функція *decision_function*, що визначає ступінь аномальності поточного стану.

Крім роботи алгоритмів аномалій, у програмну реалізацію інтегровано модуль короткострокового прогнозування. Прогноз формується за допомогою експоненційного згладжування та дозволяє оцінити очікувану динаміку показників у найближчі хвилини. Значення передаються на дашборд і відображаються у текстовому блоці «Прогноз стану», що значно підвищує інформативність системи.

Не менш важливою складовою реалізації є інтерфейс SmartHealth Dashboard. У ході експерименту тестувалася коректність роботи механізмів формування графіків, плавність оновлення даних, відсутність затримок при збільшенні обсягу історії, а також точність блоків пояснюваної аналітики, які автоматично формують текстові інтерпретації внесків ознак, рівня невизначеності та загальної оцінки стану.

3.3 Сценарії тестування

Тестування також проводилося на рівні окремих компонентів, включаючи перевірку алгоритмів аналізу, тестування API, перевірку коректності взаємодії з базою даних, відображення графіків та відгук інтерфейсу у реальному часі. Також виконано навантажувальні тести для перевірки здатності системи працювати при підвищеній кількості запитів на секунду. Значну увагу приділено валідації аналітичних алгоритмів, перевірці межових значень та симуляції аномальних станів (Таблиця 3).

Таблиця 3 - Сценарії тестування та валідації SmartHealth

№	Назва сценарію	Опис тестової ситуації	Вхідні дані (симульовані показники)	Очікувана реакція SmartHealth	Результат успішного проходження тесту
1	Перевірка роботи в нормальному стані	Тривалий стабільний стан користувача без патологій	HR: 55–85 bpm, SpO_2: 96–99%, Temp: 36.3–36.9° C	Статус: OK; низький AI-score; низькі Z-scores; жодного Alert	Система фіксує «OK», графіки рівні, відсутні тривоги
2	Тахікардія (короткочасний сплеск)	Різке підвищення ЧСС протягом 10–20 сек	HR: 145–160 bpm; інші параметри норма	Статус: ALERT (low або medium); внесок ознак → HR	Система коректно визначає HR як причину тривоги
3	Тахікардія (тривала)	Пульс утримується на підвищеном	HR: 140–170 bpm	Статус: ALERT (medium/high); високий Z-score для	Довготривала тахікардія розпізнана як серйозна аномалія

		у рівні > 60 сек		HR; AI-score негативний	
4	Гіпоксія / зниження сатурації	Критичне падіння SpO_2	SpO_2: 88–89% стабільно, HR нормальний	Статус: ALERT (high); жорстке правило SpO_2 < 90	Система негайно подає тривогу незалежно від AI-моделі
5	Висока температура	Підвищення температури до фебрильної	Temp: 38.5–39.0° C	Статус: ALERT (high); жорстке правило Temp≥38.5°C	Температурна аномалія розпізнана гарантовано
6	Шум/артефакти сенсорів	Раптові стрибки значень без фізіологічного обґрунтування	HR: 0 → 200 → 70; SpO_2: 50 → 99; нерівномірні	Модель повинна їх ігнорувати (unstable data), Uncertainty: high	Система не видає помилкових Alert, AI-score нестабільний
7	Монотонні повільні відхилення	Повільне зростання температури в межах норми	Temp: 36.4 → 37.2°C за 20 хв	OK; Z-scores низькі; аналіз стабільний	Система не спрацьовує передчасно
8	Перевірка AI-lite режиму (без sklearn)	Відсутність scikit-learn	Усі параметри у межах норми	Статус OK, працюють тільки Z-score та Mahalanobis	AI-lite працює повністю стабільно
9	Персоналізоване тренування моделі	Користувач накопичив > 10 хв даних	Нормальні дані протягом 600 сек	Кнопка «Перетренувати модель» активна; модель створена	Після тренування IF дає стабільні рішення
10	Некоректні дані або відсутність даних	База даних порожня або збої	-	Uncertainty: high; статус OK; повідомлення у Z-scores	Система коректно обробляє відсутність даних
11	Різка комбінація аномалій	HR високий, SpO_2	HR 150, SpO_2 89,	ALERT high; Explain: всі	Коректне визначення

		низький, Temp висока	Temp 38.7°C	ознаки дають внесок	мультианомалії
12	Затримка даних / повільний потік	Дані надходять нерегулярно	Інтервали між точками 3–8 сек	Графіки відображаються плавно, Uncertainty medium	UI не «ламається», система адаптується
13	Перевантаження / велика кількість даних	Велика історія вимірювань (годинна)	3600+ точок	Даунсемплінг працює, графіки не «зависають»	Інтерфейс плавний, система не падає
14	Перевірка Explainable AI	Всі параметри трохи змінені	Легкі відхилення всіх показників	Explain повинен коректно розподілити внесок ознак	Внесок ознак відповідає реальним змінам
15	Відновлення після тривалих аномалій	Після тривоги параметри повертаються до норми	HR: 150 → 80 bpm, SpO_2: 89 → 97%	Статус знову OK через 60 сек стабільності	Система не «застряє» в ALERT

Сукупність наведених механізмів формує універсальну платформу моніторингу стану здоров’я, що поєднує простоту використання, високу інформативність, гнучкість та адаптивність до конкретного користувача. Завдяки своїй архітектурі SmartHealth може бути розширений новими типами сенсорів, показників та алгоритмів, інтегрований із зовнішніми медичними сервісами або перетворений у масштабовану хмарну систему.

ВИСНОВКИ

У роботі вирішено комплексну задачу створення персоналізованої системи моніторингу стану здоров’я SmartHealth, орієнтованої на безперервний контроль життєво важливих показників та виявлення відхилень від індивідуальної норми у режимі, наближеному до реального часу.

Проведено аналіз сучасних рішень у сфері e-Health, носимих сенсорів, систем телемоніторингу та алгоритмів виявлення аномалій. Виявлено ключові обмеження існуючих продуктів, зокрема недостатню персоналізацію моделей, залежність від хмарних сервісів, непрозорість алгоритмів та складність інтеграції. На основі цього обґрунтовано доцільність розробки компактної локальної системи, що навчається безпосередньо на даних користувача та не потребує передавання інформації на сторонні сервери.

У теоретичній частині систематизовано фізіологічні показники (частота серцевих скорочень, сатурація та температура), принципи роботи відповідних сенсорів та особливості обробки часових рядів. Розглянуто методи аналізу фізіологічних даних - від класичних статистичних підходів (z-score, MAD) до моделей машинного навчання (Isolation Forest, метрика Махаланобіса). Показано, що для персональних рішень найбільш ефективним є поєднання робастної статистики та легких моделей без учителя.

Практичним результатом роботи стало створення повноцінного прототипу SmartHealth, що включає модуль емулювання сенсорних даних, локальну базу SQLite, бекенд на FastAPI, аналітичне ядро та інтерактивний дашборд. Система підтримує дві моделі роботи: статистичний «AI-lite» режим та персоналізоване навчання Isolation Forest за бажанням користувача. Реалізовано текстові пояснення з оцінкою внеску ознак, рівня невизначеності та узагальненою гісторією відхилень, що робить результати інтуїтивно зрозумілими.

Здійснено тестування системи на різних сценаріях (тахікардія, падіння сатурації, підвищення температури), перевірено коректність зберігання,

обробки та візуалізації даних, а також стабільність роботи алгоритмів виявлення аномалій.

Наукова новизна полягає в поєднанні персоналізованого аналізу, локальної обробки даних, адаптивних інтелектуальних моделей та пояснюваності результатів. Практична цінність полягає у створенні прототипу, який може стати основою для подальшої розробки медичних помічників, навчальних стендів або модулів для телемедицини.

Узагальнюючи, поставлена мета - розробити персоналізовану систему моніторингу стану здоров’я SmartHealth - виконана повністю. Реалізоване рішення відповідає сучасним тенденціям цифрової медицини, є функціональним, інтерпретованим і готовим до подальшої розробки, розширення та практичного впровадження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бойко А. В. Методи обробки інформації з відкритих джерел у контексті інформаційної безпеки: автореф. дис. ... канд. техн. наук. - К., 2021

2. OpenAI. GPT - 4 Technical Report [Електронний ресурс]. - Режим доступу: https://arxiv.org/abs/2303.08774

3. Голубєва В., Науменко Н. Ринок праці майбутнього та вплив на нього штучного інтелекту // *Цифрова екосистема сучасного університету: матеріали науково - методичної конференції*. Київ: КНЕУ, 2024. С. 45 - 49.

4. Mohamed Abdelbasset Aliouane; Jean-Marc Conrat; Jean-Christophe Cousin; Xavier Begaud Material Reflection Measurements in Centimeter and Millimeter Wave ranges for 6G Wireless Communications. - Режим доступу: https://ieeexplore.ieee.org/document/9815763

5. Klym Halyna, Kostiv Yuriy Development of multifunctional sensor system for environmental and medical applications - Режим доступу: https://ena.lpnu.ua/items/22c6555f-6a67-4329-a7d9-38cbb2d468fa

6. Liu F. T., Ting K. M., Zhou Z.-H. Isolation Forest - Режим доступу: https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf

7. QMRTools: a Mathematica toolbox for quantitative MRI analysis. - Режим доступу: DOI: 10.21105/joss.01204

8. The Impact of Artificial Intelligence on Human Fitness and the Future of Wellness. - Режим доступу: https://www.researchgate.net/publication/375113313

9. Bunker R. P., Thabtah F. A machine learning framework for sport result prediction. - Режим доступу: https://www.researchgate.net/publication/31993

10. AI-Powered Personal Fitness Coach Using Deep Learning. – Режим доступу:

https://www.internationaljournalssrg.org/IJCSE/2025/Volume12-Issue6/IJCSE-V12I6P101.pdf

11. Кабінет Міністрів України. План заходів з реалізації Концепції розвитку штучного інтелекту в Україні на 2021 - 2024 роки: Розпорядження №438 - р від 02.06.2021. Режим доступу: https://zakon.rada.gov.ua/laws/show/438 - 2021 - p

12. Makridakis S., Spiliotis E., Assimakopoulos V. The M4 Competition. Режим доступу:https://www.sciencedirect.com/science/article/pii/S0169207019301128

13. An Overview of Heart Rate Variability Metrics and Norms // PMC. 2017. Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC5624990/

14. LeCun Y., Bengio Y., Hinton G. Deep learning. 2015. Режим доступу: https://www.nature.com/articles/nature14539

15. Impact of artifact correction methods of RR series on HRV parameters // Journal of Applied Physiology. 2018. Режим доступу: https://journals.physiology.org/doi/full/10.1152/japplphysiol.00927.2016

16. Automatic Near Real-Time Outlier Detection and Correction in Cardiac Interbeat Interval Series for HRV Analysis // JMIR Biomedical Engineering. 2019. Режим доступу: https://www.researchgate.net/publication/330761958

17. Скрипник Юрій Васильович. Система моніторингу здоров'я пацієнтів на основі IoT з використанням ESP8266 і Arduino. Режим доступу: https://ce-college.chnu.edu.ua/media/htyc5gne/skrypnyk.pdf

18. Євгеній Перетяка. Алгоритми виявлення критичних станів здоров’я людини на основі аналізу фізіологічних та візуальних індикаторів. Режим доступу: https://openarchive.nure.ua/server/api/core/bitstreams/b4e2c0aa-3643-4e22-b614-8c7d4ae7d79e/content

19. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. 2020.ї

20. Validation of nocturnal resting heart rate and heart rate variability in consumer wearables. 2025. Режим доступу: https://pubmed.ncbi.nlm.nih.gov/40834291/

21. Стрельніков, В. І., Бондарчук, А. П. (2025). Комплексний підхід до інтелектуального управління, моделювання та виявлення мережних аномалій на основі ентропійних та нейромережевих підходів. Телекомунікаційні та інформаційні технології, (2), 100-107. DOI 10.31673/2412-4338.2025.025703

22. Тетяна Носенко, Ірина Машкіна (2025) Моделювання процесів обробки екологічних даних для систем мобільного моніторингу на основі БПЛА та методу IDW Електронне фахове наукове видання «Кібербезпека: освіта, наука, техніка», No 2 (30), 2025 с.110-122. Спеціальний випуск ISSN 2663 – 4023 https://doi.org/10.28925/2663-4023.2025.30.955

23. Бушма, Олександр Володимирович та Машкіна, Ірина Вікторівна та Носенко, Тетяна Іванівна та Яскевич, Владислав Олександрович (2024) *Кваліфікаційна робота магістра: Навчально-методичний посібник для спеціальності «Комп'ютерні науки»* Київський столичний університет імені Бориса Грінченка, Україна.https://elibrary.kubg.edu.ua/id/eprint/50205/

Додаток А

Лістинг програмної реалізації

backend.py > ...

```python
import time, sqlite3
from pathlib import Path
from typing import List, Dict, Optional

import numpy as np
import pandas as pd
from fastapi import FastAPI, Query
from fastapi.responses import HTMLResponse, JSONResponse
from pydantic import BaseModel, Field
from joblib import dump, load
import json, os
from datetime import datetime

TRAIN = {}  # per-user: {'state': 'idle|training|done|error', 'msg': str, 'started_at': ts, 'finished_at': ts}
# спроба імпорту sklearn; якщо нема — працюємо в AI-lite режимі
try:
    from sklearn.ensemble import IsolationForest
    HAVE_SKLEARN = True
except Exception:
    HAVE_SKLEARN = False
    IsolationForest = None  # type: ignore

DB_PATH = "smarthealth.db"
MODEL_DIR = Path("model_store"); MODEL_DIR.mkdir(exist_ok=True)

app = FastAPI(title="SmartHealth Backend")

def init_db():
    with sqlite3.connect(DB_PATH) as con:
        con.execute("""
        CREATE TABLE IF NOT EXISTS vitals (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            ts REAL NOT NULL,
            hr REAL,
            spo2 REAL,
            temp REAL,
            user_id TEXT DEFAULT 'demo'
        );
        """)
init_db()

# ---------- Pydantic моделі ----------
class Sample(BaseModel):
    ts: float = Field(default_factory=lambda: time.time())
    hr: float
    spo2: float
    temp: float
    user_id: str = "demo"   # 👈 персоналізація
```

backend.py > ...

```python

class IngestBatch(BaseModel):
    data: List[Sample]

class AnomalyResponse(BaseModel):
    status: str
    severity: str
    reason: Dict[str, float]
    zscores: Dict[str, float]
    score: float
    uncertainty: str
    forecast: Dict[str, float]

# ---------- допоміжні ----------
def model_path_for(user_id: str) -> Path:
    return MODEL_DIR / f"{user_id}_iforest.joblib"

def meta_path_for(user_id: str) -> Path:
    return MODEL_DIR / f"{user_id}_iforest_meta.json"

def save_meta(user_id: str, meta: dict):
    meta["user_id"] = user_id
    meta["trained_at_iso"] = datetime.utcnow().isoformat() + "Z"
    meta_path_for(user_id).write_text(json.dumps(meta, ensure_ascii=False, indent=2), encoding="utf-8")

def load_meta(user_id: str) -> Optional[dict]:
    p = meta_path_for(user_id)
    return json.loads(p.read_text(encoding="utf-8")) if p.exists() else None

def insert_samples(samples: List[Sample]):
    rows = [(s.ts, s.hr, s.spo2, s.temp, s.user_id) for s in samples]
    with sqlite3.connect(DB_PATH) as con:
        con.executemany(
            "INSERT INTO vitals (ts, hr, spo2, temp, user_id) VALUES (?, ?, ?, ?, ?);", rows
        )

def load_last_minutes(user_id: str, minutes: float = 30.0) -> pd.DataFrame:
    tmin = time.time() - minutes * 60.0
    with sqlite3.connect(DB_PATH) as con:
        df = pd.read_sql_query(
            "SELECT ts, hr, spo2, temp FROM vitals WHERE ts >= ? AND user_id=? ORDER BY ts ASC;",
            con, params=(tmin, user_id)
        )
    return df

def load_all(user_id: str) -> pd.DataFrame:
    with sqlite3.connect(DB_PATH) as con:
        df = pd.read_sql_query(
```

backend.py > ...

```python
def load_all(user_id: str) -> pd.DataFrame:
    with sqlite3.connect(DB_PATH) as con:
        df = pd.read_sql_query(
            "SELECT ts, hr, spo2, temp FROM vitals WHERE user_id=? ORDER BY ts ASC;",
            con, params=(user_id,)
        )
    return df

def rolling_baseline(series: pd.Series, window: int = 60):
    med = series.rolling(window=window, min_periods=max(5, window//3)).median()
    mad = (series.rolling(window=window, min_periods=max(5, window//3))
           .apply(lambda x: np.median(np.abs(x - np.median(x))), raw=True))
    sigma = 1.4826 * mad
    return med, sigma.replace(0, np.nan)

def zscore_last_point(df: pd.DataFrame, window: int = 60) -> Dict[str, float]:
    z = {}
    for col in ["hr", "spo2", "temp"]:
        med, sigma = rolling_baseline(df[col], window=window)
        x = df[col].iloc[-1]
        m = med.iloc[-1] if not np.isnan(med.iloc[-1]) else df[col].median()
        s = sigma.iloc[-1] if not np.isnan(sigma.iloc[-1]) else (df[col].std() or 1.0)
        z[col] = float((x - m) / (s if s != 0 else 1.0))
    return z

def expo_forecast(series, alpha=0.5):
    # series — pandas.Series
    if len(series) < 2:
        return float(series.iloc[-1])
    forecast = float(series.iloc[0])
    for v in series:
        forecast = alpha * float(v) + (1 - alpha) * forecast
    return forecast

def extract_features(df: pd.DataFrame, window_secs: int = 60) -> Optional[np.ndarray]:
    if len(df) < window_secs: return None
    win = df.iloc[-window_secs:]
    feats = []
    for col in ["hr", "spo2", "temp"]:
        x = win[col].dropna().to_numpy()
        if len(x) == 0: x = np.array([np.nan])
        s = pd.Series(x)
        feats.extend([
            float(np.nanmean(x)), float(np.nanstd(x)), float(s.skew()), float(s.kurt()),
            float(np.nanmin(x)), float(np.nanmax(x)), float(x[-1]-x[0])
        ])
    return np.array(feats, dtype=float).reshape(1, -1)
```

backend.py > ...

```python

def train_iforest(df: pd.DataFrame):
    if not HAVE_SKLEARN:
        raise RuntimeError("scikit-learn недоступний: тренування IF пропущено.")
    if len(df) < 10*60:
        raise ValueError("Недостатньо даних для тренування (менше 10 хв).")
    X = []
    step = 5
    for end in range(60, len(df), step):
        feats = extract_features(df.iloc[:end], window_secs=60)
        if feats is not None: X.append(feats[0])
    X = np.array(X)
    model = IsolationForest(n_estimators=200, contamination="auto", random_state=42)
    model.fit(X)
    return model

def decision_iforest(model, feats: np.ndarray) -> float:
    return float(model.decision_function(feats)[0])

# ---- AI-lite ----
def mahalanobis_score(X_hist: np.ndarray, x_now: np.ndarray) -> float:
    mu = np.nanmean(X_hist, axis=0)
    Xc = X_hist - mu
    cov = np.cov(np.nan_to_num(Xc, nan=0.0).T) + 1e-6*np.eye(Xc.shape[1])
    try:
        inv = np.linalg.inv(cov)
    except np.linalg.LinAlgError:
        inv = np.linalg.pinv(cov)
    d = x_now - mu
    return float(np.sqrt(d @ inv @ d.T))

def build_hist_features(df: pd.DataFrame, minutes: int = 10) -> Optional[np.ndarray]:
    if len(df) < minutes*60: return None
    X = []
    step = 5
    for end in range(60, min(len(df), minutes*60)+1, step):
        feats = extract_features(df.iloc[:end], window_secs=60)
        if feats is not None: X.append(feats[0])
    return np.array(X) if len(X) else None

def hard_rules(last: Dict[str, float]) -> Optional[str]:
    if last["spo2"] < 90: return "SpO2<90%"
    if last["temp"] >= 38.5: return "Temp>=38.5"
    if last["hr"] >= 140: return "HR>=140"
    return None

# ---------- API ----------
@app.get("/health")
```

backend.py > ...

```python
@app.get("/health")
def health(): return {"ok": True, "sklearn": HAVE_SKLEARN}

@app.post("/ingest")
def ingest(batch: IngestBatch):
    insert_samples(batch.data)
    return {"accepted": len(batch.data)}

@app.post("/train")
def train(user_id: str = Query("demo")):
    if not HAVE_SKLEARN:
        return {"trained": False, "reason": "scikit-learn недоступний; використовуйте AI-lite"}

    df = load_all(user_id)
    TRAIN[user_id] = {"state": "training", "msg": "Збір даних…", "started_at": time.time(), "finished_at": None}

    try:
        if len(df) < 10*60:  # ~10 хв при 1 точці/сек
            TRAIN[user_id] = {"state":"error","msg":"Недостатньо даних (<10 хв).","started_at":TRAIN[user_id]["started_at"],"finished_at":time.time()}
            return {"trained": False, "reason": "Недостатньо даних для тренування (менше 10 хв)."}

        # будуємо навчальну вибірку з ковзних вікон
        TRAIN[user_id]["msg"] = "Побудова ознак…"
        feature_names = []
        for col in ["hr","spo2","temp"]:
            feature_names += [
              f"{col}_mean", f"{col}_std",  f"{col}_skew",
              f"{col}_kurt", f"{col}_min",  f"{col}_max", f"{col}_delta"
            ]

        X = []
        step = 5
        for end in range(60, len(df), step):
            feats = extract_features(df.iloc[:end], window_secs=60)
            if feats is not None: X.append(feats[0])
        X = np.array(X)

        TRAIN[user_id]["msg"] = f"Навчання IsolationForest на {len(X)} вікнах…"
        model = IsolationForest(n_estimators=200, contamination="auto", random_state=42)
        model.fit(X)

        dump(model, model_path_for(user_id))
        meta = {
            "n_windows": int(len(X)),
            "window_secs": 60,
            "step_secs": step,
            "feature_names": feature_names,
            "n_estimators": 200,
```

backend.py > ...

```python
def train(user_id: str = Query("demo")):
                "contamination": "auto",
                "time_range": {"first_ts": float(df["ts"].iloc[0]), "last_ts": float(df["ts"].iloc[-1])},
            }
            save_meta(user_id, meta)

            TRAIN[user_id] = {"state":"done","msg":"Готово","started_at":TRAIN[user_id]["started_at"],"finished_at":time.time()}
            return {"trained": True, "samples": int(len(df)), "user_id": user_id, "n_windows": int(len(X))}
        except Exception as e:
            TRAIN[user_id] = {"state":"error","msg":str(e),"started_at":TRAIN[user_id]["started_at"],"finished_at":time.time()}
            return {"trained": False, "reason": f"{e.__class__.__name__}: {e}"}

@app.get("/train_status")
def train_status(user_id: str = Query("demo")):
    st = TRAIN.get(user_id, {"state":"idle","msg":"—","started_at":None,"finished_at":None})
    meta = load_meta(user_id)
    have_model = model_path_for(user_id).exists()
    return {
        "status": st,
        "model_exists": have_model,
        "meta": meta
    }

@app.get("/data_stats")
def data_stats(user_id: str = Query("demo"), minutes: int = Query(30)):
    df = load_last_minutes(user_id, minutes)
    return {
        "rows": int(len(df)),
        "minutes": minutes,
        "first_ts": float(df["ts"].iloc[0]) if len(df) else None,
        "last_ts": float(df["ts"].iloc[-1]) if len(df) else None
    }

@app.get("/latest", response_model=AnomalyResponse)
def latest(user_id: str = Query("demo")):
    df = load_last_minutes(user_id, 30)
    if len(df) == 0:
        return AnomalyResponse(
            status="ok",
            severity="low",
            reason={},
            zscores={},
            score=0.0,
            uncertainty="high",
            forecast={"hr": None, "spo2": None, "temp": None}
        )

    forecast_hr = expo_forecast(df["hr"])
```

backend.py > ...

```python
def latest(user_id: str = Query("demo")):
    forecast_spo2 = expo_forecast(df["spo2"])
    forecast_temp = expo_forecast(df["temp"])

    # останні значення та жорсткі правила
    last = df.iloc[-1][["hr", "spo2", "temp"]].to_dict()
    violated = hard_rules(last)

    # персональні z-score
    z = zscore_last_point(df, window=60)
    abs_z = {k: abs(v) for k, v in z.items()}
    denom = sum(abs_z.values()) or 1.0
    explain = {k: round(v / denom, 3) for k, v in abs_z.items()}

    # фічі для IF/AI-lite
    feats_now = extract_features(df, window_secs=60)
    score = 0.0
    uncertainty = "low"
    used_iforest = False  # чи реально використали треновану IF-модель

    if feats_now is not None:
        mpath = model_path_for(user_id)
        if HAVE_SKLEARN and mpath.exists():
            # ► використаємо Isolation Forest
            model = load(mpath)
            score = decision_iforest(model, feats_now)
            used_iforest = True
            # шкала невизначеності для IF
            if -0.05 <= score <= 0.05:
                uncertainty = "high"
            elif -0.15 <= score <= 0.15:
                uncertainty = "medium"
            else:
                uncertainty = "low"
        else:
            # ► AI-lite: історичні фічі + Махаланобіс (score – лише індикатор)
            X_hist = build_hist_features(df, minutes=10)
            if X_hist is not None and X_hist.shape[0] >= 6:
                d = mahalanobis_score(X_hist, feats_now.reshape(-1))
                score = float(-d)  # показуємо у UI, але не використовуємо для рішення
                if d < 2.0:
                    uncertainty = "low"
                elif d < 3.0:
                    uncertainty = "medium"
                else:
                    uncertainty = "high"
            else:
                score, uncertainty = 0.0, "high"
```

backend.py > ...

```python
def latest(user_id: str = Query("demo")):
            # гілка для Isolation Forest
            if score < -0.1 and (abs(z["hr"]) > 3 or abs(z["spo2"]) > 3 or abs(z["temp"]) > 3):
                status, severity = "alert", "medium"
            elif score < -0.2:
                status, severity = "alert", "low"
            else:
                status, severity = "ok", "low"
        else:
            # чистий AI-lite: рішення ТІЛЬКИ за z-score
            zsum = abs(z["hr"]) + abs(z["spo2"]) + abs(z["temp"])
            if zsum > 8 or (abs(z["spo2"]) > 3.5):
                status, severity = "alert", "medium"
            elif zsum > 5:
                status, severity = "alert", "low"
            else:
                status, severity = "ok", "low"

    return AnomalyResponse(
        status=status,
        severity=severity,
        reason=explain,
        zscores={k: round(v, 2) for k, v in z.items()},
        score=round(float(score), 3),
        uncertainty=uncertainty,
        forecast={
        "hr": round(forecast_hr, 2),
        "spo2": round(forecast_spo2, 2),
        "temp": round(forecast_temp, 2)
        }
    )
@app.get("/users")
def users():
    with sqlite3.connect(DB_PATH) as con:
        rows = con.execute("SELECT DISTINCT user_id FROM vitals ORDER BY user_id").fetchall()
    return JSONResponse([r[0] for r in rows])

# ---- часові ряди для графіків ----
@app.get("/series")
def series(user_id: str = Query("demo"), minutes: int = Query(30)):
    df = load_last_minutes(user_id, minutes)
    out = {
        "ts": df["ts"].tolist(),
        "hr": df["hr"].tolist(),
        "spo2": df["spo2"].tolist(),
        "temp": df["temp"].tolist()
    }
    return JSONResponse(out)
```